

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表平8-500200

(43) 公表日 平成8年(1996)1月9日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I		
G 0 6 F 17/21					
3/14	3 5 0 A	7323-5B			
9/06	4 1 0 S	7230-5B			
		9288-5L	G 0 6 F 15/20	5 7 0 D	
		9288-5L		5 9 6 B	
			審査請求 有	予備審査請求 有	(全 126 頁) 最終頁に続く

(21) 出願番号 特願平6-514206
(86) (22) 出願日 平成5年(1993)11月24日
(85) 翻訳文提出日 平成7年(1995)5月30日
(86) 国際出願番号 PCT/US93/11468
(87) 国際公開番号 WO94/14115
(87) 国際公開日 平成6年(1994)6月23日
(31) 優先権主張番号 07/984, 868
(32) 優先日 1992年12月1日
(33) 優先権主張国 米国 (US)
(81) 指定国 EP (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, M C, NL, PT, SE), CA, JP

(71) 出願人 マイクロソフト コーポレーション
アメリカ合衆国 ワシントン州 98052 -
6399 レッドモンド ワン マイクロソフ
ト ウェイ (番地なし)
(72) 発明者 コッポル スリニヴァサ アール
アメリカ合衆国 ワシントン州 98052
レッドモンド トゥーハンドレッドアンド
サーティシックス アベニュー ノース
イースト 2402
(74) 代理人 弁理士 中村 稔 (外7名)

最終頁に続く

(54) 【発明の名称】 埋め込まれたオブジェクトとイン・プレーce対話するための方法及びシステム

(57) 【要約】

コンテナオブジェクト内に収容されたコンテナオブジェクトと対話するためのコンピュータ方法及びシステムが提供される。本発明の好ましい実施例において、コンテナオブジェクトは、該コンテナオブジェクトと対話するためのコンテナリソースを有するコンテナウインドウ環境を伴うコンテナアプリケーションを有している。コンテナオブジェクトは、該コンテナオブジェクトと対話するためのサーバリソースをもつサーバウインドウ環境を伴うサーバアプリケーションを有している。本発明の方法は、コンテナウインドウ環境をディスプレイ装置に表示する。次いで、ユーザはコンテナオブジェクトを選択する。コンテナオブジェクトの選択に回答して、この方法は、複数のサーバリソースを、表示されたコンテナウインドウ環境と一体化する。ユーザがサーバリソースを選択したときに、この方法は、サーバアプリケーションを呼び出して、サーバリソース選択を処理する。これに対し、ユーザがコンテナリソースを選択したときは、この方法は、コンテナアプリケーションを呼び出して、そのコンテナリソ

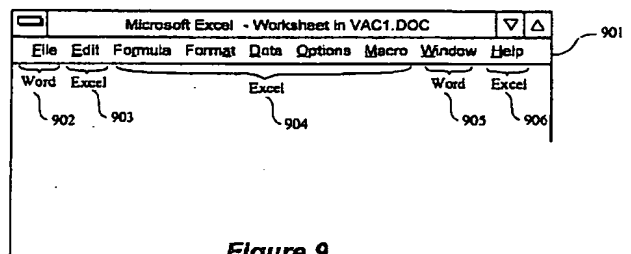


Figure 9

【特許請求の範囲】

1. コンピュータシステムにおいてコンテナオブジェクト内に収容されたコンテナオブジェクトをアクチベートするための方法であって、コンテナオブジェクトは、コンテナウインドウ環境を伴うコンテナアプリケーションを有し、コンテナウインドウ環境は、コンテナオブジェクトと対話するためのコンテナリソースを有し、コンテナオブジェクトは、該コンテナオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションを有し、上記方法は、

上記コンテナウインドウ環境を表示し、

上記表示されたコンテナウインドウ環境内に上記コンテナオブジェクトを表示し、

上記コンテナオブジェクトを選択し、そして

複数のサーバリソースを上記表示されたコンテナ環境と一体化して、ユーザがサーバリソースを選択するときに、サーバアプリケーションがサーバリソースの選択を処理するようにする、

という段階を備えたことを特徴とする方法。

2. 上記コンテナアプリケーションはコンテナメニューを有し、上記サーバアプリケーションはサーバメニューを有し、複数のサーバリソースを一体化する上記段階は、サーバメニュー及びコンテナメニューを有する複合メニューバーを発生する請求項1に記載の方法。

3. 上記一体化段階は、サーバメニュー及びコンテナメニューを複合メニューバーにおいてインターリーブする請求項2に記載の方法。

4. 上記コンテナアプリケーションは、複数のメニューを表示するためのメニューバーを有し、そして上記複合メニューバーは、コンテナアプリケーションのメニューバーとして表示される請求項2に記載の方法。

5. コンピュータシステムにおいてコンテナオブジェクト内に収容されたコンテナオブジェクトと対話するための方法であって、コンテナオブジェクトは、コンテナウインドウ環境を伴うコンテナアプリケーションを有し、コンテナウインドウ環境は、コンテナオブジェクトと対話するためのコンテナ

リソースを有し、コンテニーオブジェクトは、サーバウインドウ環境を伴うサーバアプリケーションを有し、サーバウインドウ環境は、コンテニーオブジェクトと対話するためのサーバリソースを有し、上記方法は、

上記コンテナウインドウ環境を表示し、

上記コンテニーオブジェクトを選択し、

複数のサーバリソースを上記コンテナウインドウ環境と一体化し、

ユーザがサーバリソースを選択するときには、そのサーバリソース選択を処理するようにサーバアプリケーションを呼び出し、そして

ユーザがコンテナリソースを選択するときには、そのコンテナリソース選択を処理するようにコンテナアプリケーションを呼び出す、

という段階を備えたことを特徴とする方法。

6. 上記コンテナアプリケーションは、データを表示するためのウインドウを有し、そしてそのウインドウ内にサーバリソースを配置するようにコンテナアプリケーションと交渉する段階を備えた請求項5に記載の方法。

7. サーバアプリケーションのみがサーバリソースの知識を有する請求項6に記載の方法。

8. 上記コンテナアプリケーション及びサーバアプリケーションは、個別のコンピュータプロセスとして実行される請求項5に記載の方法。

9. 上記コンテニーオブジェクトを表示し、そしてその表示されたコンテニーオブジェクトをハイライト処理して、サーバリソースがユーザ選択に使用できることを指示する段階を備えた請求項5に記載の方法。

10. 上記コンテナアプリケーションは、メッセージを受け取って処理するためのコンテナメッセージハンドラーを有し、上記サーバアプリケーションは、メッセージを受け取って処理するためのサーバメッセージハンドラーを有し、そして上記方法は、更に、上記コンテナメッセージハンドラーを特殊なメッセージハンドラーと置き換える段階を備え、この特殊なメッセージハンドラーは、コンテナリソース選択メッセージをコンテナメッセージハンドラーへ送ると共に、サーバリソース選択メッセージをサーバメッセージハンドラーへ送るものである請求項5に記載の方法。

11. 上記コンテナアプリケーションはウインドウを有し、上記サーバアプリケーションはウインドウを有し、そして更に、

上記サーバアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

ユーザからメニューコマンドを受け取り、そして

上記メニューコマンドを受け取るのに応答して、

上記コンテナアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

メニューニューモニックを受け取り、そして

メニューニューモニックを受け取るのに応答して、サーバアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして再指定する、

という段階を含む請求項5に記載の方法。

12. 上記コンテナアプリケーションのウインドウは、フレームウインドウである請求項11に記載の方法。

13. 上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナアプリケーションは、コンテナリソースを選択するための複数のアクセラレータキー組合せを有し、上記サーバアプリケーションは、サーバリソースを選択するための複数のアクセラレータキー組合せを有し、そして上記コンテナアプリケーションは、アクセラレータキー組合せを受け取るときに、サーバアプリケーションを呼び出して、サーバリソースが選択されたかどうか決定する請求項5に記載の方法。

14. コンピュータシステムにおいてコンテナオブジェクト内に収容されたコンテナオブジェクトと対話するための方法であって、コンテナオブジェクトは、複数のコンテナメニューを伴う関連するコンテナアプリケーションを有し、このコンテナアプリケーションは、メニューのリストを表示するための関連するメニューバーを有し、コンテナオブジェクトは、複数のサーバメニューを伴う関連するサーバアプリケーションを有し、上記方法は、

コンテナメニュー及びサーバメニューを含む複合メニューリストを発生し、

上記複合メニューリストをメニューバーに表示し、
表示されたコンテナメニューをユーザが選択するのに応答して、コンテナアプリケーションを呼び出してその選択されたメニューを処理し、そして
表示されたサーバメニューをユーザが選択するのに応答して、サーバアプリケーションを呼び出してその選択されたメニューを処理する、
という段階を備えたことを特徴とする方法。

15. コンピュータシステムにおいて複数のアプリケーションからのメニューを一体化する方法であって、

各アプリケーションごとに1組のメニューグループを定義し、各メニューグループは複数のメニュー項目を有し、

上記メニューグループを複合メニューに結合し、
複合メニューをディスプレイ装置に表示し、
表示された複合メニューのメニューグループのメニュー項目を選択し、そして
その選択されたメニュー項目のメニューグループを定義するアプリケーションを呼び出す、

という段階を備えたことを特徴とする方法。

16. 上記複数のアプリケーションは、コンテナアプリケーション及びサーバアプリケーションを含み、そして複合メニューを表示する上記段階は、複合メニューをコンテナアプリケーションのメニューとして表示する請求項15に記載の方法。

17. コンピュータシステムにおいてウインドウハイアラキを動的に結合する方法であって、上記コンピュータシステムは、ディスプレイ装置と、該ディスプレイ装置におけるウインドウの表示を管理するためのウインドウシステムとを有し、上記方法は、

第1のアプリケーションプログラムを呼び出しそして第1のウインドウハイアラキを形成し、

上記第1のウインドウハイアラキのウインドウを、第2のアプリケーションプログラムの親ウインドウとして指定し、そして 第2のアプリケーションプログラムを呼び出しそして第2のウインドウハイアラキを形成し、

ラーキを形成し、ウインドウハイアラークのルートウインドウが上記指定されたウインドウの子ウインドウとなるようにする、

という段階を備えたことを特徴とする方法。

18. コンピュータシステムにおいてウインドウをスクロールする方法であって、上記コンピュータシステムは、ディスプレイ装置と、該ディスプレイ装置におけるウインドウの表示を管理するためのウインドウシステムとを有し、該ウインドウシステムは、ウインドウ内に表示されたデータをスクロールできるようにするものであり、上記方法は、

選択されるべきデータを指示し、該データは、外側のウインドウ内に収容された内側のウインドウ内に収容され、

外側のウインドウをスクロールするためのユーザ入力に応答して、上記指示されたデータを選択されたデータとして維持する、

という段階を備えたことを特徴とする方法。

19. 内側のウインドウがユーザの視野から外れてスクロールされたときに上記指示されたデータを選択されたものとして維持する段階を備えた請求項18に記載の方法。

20. 内側のウインドウを、入力を受けるための入力フォーカスを有するものとして指定し、そして内側のウインドウがユーザの視野から外れてスクロールされたときに内側のウインドウを、入力フォーカスを有するものとして指定維持するという段階を備えた請求項18に記載の方法。

21. コンピュータシステムにおいてディスプレイ装置に表示されたオブジェクトが選択されたことを指示する方法であって、陰影付けされたパターンのボーダーをオブジェクトの周りに表示する段階を備えたことを特徴とする方法。

22. コンテナオブジェクト内に収容されたコンテナオブジェクトをアクチベートするためのコンピュータシステムであって、上記コンテナオブジェクトは、コンテナウインドウ環境を伴うコンテナアプリケーションを有し、該コンテナウインドウ環境は、コンテナオブジェクトと対話するためのコンテナリソースを有し、コンテナオブジェクトは、該コンテナオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションを有し、上記シス

テムは、

上記コンテナウインドウ環境を表示する手段と、

上記表示されたコンテナウインドウ環境内に上記コンテナオブジェクトを表示する手段と、

上記コンテナオブジェクトを選択する手段と、

複数のサーバリソースを上記表示されたコンテナ環境と一体化する手段であって、ユーザがサーバリソースを選択するときに、サーバアプリケーションがサーバリソースの選択を処理するようにする手段と、
を備えたことを特徴とするコンピュータシステム。

23. 上記コンテナアプリケーションはコンテナメニューを有し、上記サーバアプリケーションはサーバメニューを有し、上記一体化手段は、サーバメニュー及びコンテナメニューを有する複合メニューバーを発生する手段を備えた請求項22に記載のシステム。

24. 上記一体化手段は、サーバメニュー及びコンテナメニューを複合メニューバーにおいてインターリーブする手段を含む請求項23に記載のシステム。

25. 上記コンテナアプリケーションは、複数のメニューを表示するためのメニューバーを有し、そして上記複合メニューバーは、コンテナアプリケーションのメニューバーとして表示される請求項23に記載のシステム。

26. コンテナオブジェクト内に収容されたコンテナオブジェクトと対話するためのコンピュータシステムであって、コンテナオブジェクトは、コンテナウインドウ環境を伴うコンテナアプリケーションを有し、コンテナウインドウ環境は、コンテナオブジェクトと対話するためのコンテナリソースを有し、コンテナオブジェクトは、サーバウインドウ環境を伴うサーバアプリケーションを有し、サーバウインドウ環境は、コンテナオブジェクトと対話するためのサーバリソースを有し、上記システムは、

上記コンテナウインドウ環境をディスプレイ装置に表示する手段と、

上記コンテナオブジェクトを選択する手段と、

コンテナオブジェクトが選択されたときに複数のサーバリソースを上記コンテナウインドウ環境と一体化する手段と、

サーバリソースを選択する手段と、

サーバリソースが選択されたときに、そのサーバリソース選択を処理するようにサーバアプリケーションを呼び出す手段と、

コンテナリソースを選択する手段と、

コンテナリソースが選択されたときに、そのコンテナリソース選択を処理するようにコンテナアプリケーションを呼び出す手段と、
を備えたことを特徴とするコンピュータシステム。

27. 上記コンテナアプリケーションは、データを表示するためのウィンドウを有し、そして上記一体化手段は、そのウィンドウ内にサーバリソースを配置するようにコンテナアプリケーションと交渉する手段を備えた請求項26に記載のシステム。

28. サーバアプリケーションのみがサーバリソースの知識を有する請求項27に記載のシステム。

29. 上記コンテナアプリケーション及びサーバアプリケーションは、個別のコンピュータプロセスとして実行される請求項26に記載のシステム。

30. 上記コンテナオブジェクトを表示する手段と、その表示されたコンテナオブジェクトをハイライト処理して、サーバリソースがユーザ選択に使用できることを指示する手段とを備えた請求項26に記載のシステム。

31. 上記コンテナアプリケーションは、メッセージを受け取って処理するためのコンテナメッセージハンドラーを有し、上記サーバアプリケーションは、メッセージを受け取って処理するためのサーバメッセージハンドラーを有し、そして上記システムは、更に、上記コンテナメッセージハンドラーを特殊なメッセージハンドラーと置き換える手段を備え、該特殊なメッセージハンドラーは、コンテナリソース選択メッセージをコンテナメッセージハンドラーへ送ると共に、サーバリソース選択メッセージをサーバメッセージハンドラーへ送るものである請求項26に記載のシステム。

32. 上記コンテナアプリケーションはウィンドウを有し、上記サーバアプリケーションはウィンドウを有し、そして更に、

上記サーバアプリケーションウィンドウを、ユーザ入力を受け取るための入力

フォーカスを有するものとして指定する手段と、

ユーザからメニューコマンドを受け取る手段と、

メニューコマンドを受け取るのに応答して、上記コンテナアプリケーションウィンドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定する手段と、

メニューニューモニックを受け取る手段と、

メニューニューモニックを受け取るのに応答して、サーバアプリケーションウィンドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして再指定する手段と、

を備えた請求項26に記載のシステム。

33. 上記コンテナアプリケーションのウィンドウは、フレームウィンドウである請求項32に記載のシステム。

34. 上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナアプリケーションは、コンテナリソースを選択するための複数のアクセラレータキー組合せを有し、上記サーバアプリケーションは、サーバリソースを選択するための複数のアクセラレータキー組合せを有し、そしてアクセラレータキー組合せを受け取る時に、サーバアプリケーションを呼び出して、サーバリソースが選択されたかどうか決定する手段を備えた請求項26に記載のシステム。

【発明の詳細な説明】

埋め込まれたオブジェクトとイン・プレース対話するための方法及びシステム

発明の分野

本発明は、一般に、リンクされそして埋め込まれたオブジェクトと対話するためのコンピュータ方法及びシステムに係り、より詳細には、コンテナアプリケーションのコンテキスト内に収容されたオブジェクトを編集し、さもなくば、それと対話するための方法及びシステムに係る。

先行技術の説明

現在の文書処理コンピュータシステムは、ユーザが複合文書を作成できるようにする。複合文書とは、種々のフォーマットの情報を含む文書である。例えば、複合文書は、テキストフォーマット、チャートフォーマット、数字フォーマット等のデータを含む。図1は、複合文書の一例である。この例において、複合文書101は、ある製造オブジェクトに対するレポートとして形成される。この複合文書101は、チャートフォーマットで表されたスケジュールデータ102と、スプレッドシートフォーマットで表された予算データ103と、テキストフォーマットで表された説明データ104とを含んでいる。典型的な公知のシステムでは、ユーザは、プロジェクトマネジメントコンピュータプログラムを用いてスケジュールデータ102を発生し、そしてスプレッドシートコンピュータプログラムを用いて予算データ103を発生する。このデータが発生された後に、ユーザは、複合文書101を形成し、説明データ104を入力し、そしてワードプロセッサコンピュータプログラムを使用してスケジュールデータ102と予算データ103を結合する。

図2は、スケジュールデータ、予算データ及び説明データを複合文書へいかに組み込むかを示している。ユーザは、プロジェクトマネジメントプログラム201を使用してスケジュールデータを発生し、そしてそのデータをクリップボード203に記憶する。ユーザは、スプレッドシートプログラム204を使用して予算データを発生し、次いで、そのデータをクリップボード203に記憶する。クリップボード203は、通常はいかなるプログラムによってもアクセスできる

記憶エリア（ディスク又はメモリ）である。プロジェクトマネジメントプログラム201及びスプレッドシートプログラム204は、通常は、データをプレゼンテーションフォーマットでクリップボードに記憶する。プレゼンテーションフォーマットとは、データが出力装置に容易に表示されるフォーマットである。例えば、プレゼンテーションフォーマットは、標準的なビットマップブロック転送動作（B i t B l t）で表示することのできるビットマップである。データをクリップボードに記憶することを、クリップボードに「コピーする」と称する。

データがクリップボード203にコピーされた後に、ユーザは、ワード処理プログラム206を始動して複合文書101を作成する。ユーザは、説明データ104を入力し、そしてクリップボード203にあるスケジュールデータ及び予算データをコピーすべき複合文書101内の位置を指定する。クリップボードから文書へデータをコピーすることを、クリップボードから「ペーストする」と称する。次いで、ワードプロセスプログラム206は、スケジュールデータ102及び予算データ103をクリップボード203から複合文書101の指定の位置へコピーする。クリップボードから複合文書へコピーされるデータは、「埋め込まれる」データと称する。ワードプロセスプログラム206は、この埋め込まれるデータを、複合文書101を出力装置においてレンダリングするときにB i t B l t動作で表示する単純なビットマップとして処理する。ある公知システムにおいては、クリップボードは、一度に1つのコピーコマンドに対してデータを記憶することしかできない。このようなシステムでは、スケジュールデータは、クリップボードにコピーし、そして複合文書へペーストすることができる。次いで、予算データをクリップボードにコピーし、そして複合文書へペーストすることができる。

ワードプロセッサは、通常、テキストデータしか処理しないので、ワードプロセスプログラムのユーザは、埋め込まれるデータを移動又は削除することはできないが、埋め込まれるデータがテキストフォーマットでない限りそれを変更することはできない。従って、ユーザが、例えば、複合文書101内にある予算データ103を変更したい場合には、ユーザは、スプレッドシートプログラム204をスタートさせ、予算データ103をファイルからロードし、変更を行い、その

変更をクリップボード203にコピーし、ワードプロセスプログラム206をスタートさせ、複合文書101をロードし、そして変更されたクリップボードデータを複合文書101へペーストする。スプレッドシートプログラムは、スプレッドシートデータを「具現化」し、即ちスプレッドシートプログラムを用いてスプレッドシートフォーマットのデータを操作することができる。プログラムが具現化するフォーマットを「ネイティブ」フォーマットと称する。

公知のシステムは、データを実際に埋め込むのではなく、複合文書に含まれるべきデータに対するリンクを記憶する。ワードプロセスプログラムがデータをクリップボードから複合文書へペーストするときには、複合文書にリンクが記憶される。このリンクは、含まれるべきデータ（通常はファイルに存在する）を指すものである。これらの公知システムは、通常、データに対するリンクを、ワードプロセスプログラムがプレゼンテーションフォーマットとして確認又は処理するフォーマットで与える。例えば、ワードプロセスプログラム206が、ユーザにより、スケジュールデータ及び予算データを複合文書へ埋め込みではなくてリンクによってペーストするように指示されるときには、スケジュールデータ及び予算データがプレゼンテーションフォーマットで存在するファイルの名前が文書に挿入される。データの1つのコピーを多数の複合文書により共用できるようにするために、多数の複合文書が同じデータに対するリンクを含むことができる。

発明の要旨

本発明の目的は、コンテナ（収容体）オブジェクトのコンテナアプリケーションのウィンドウ環境内に収容されたオブジェクトと対話するための方法及びシステムを提供することである。

本発明の別の方法は、コンテナアプリケーションのメニューを、収容されたオブジェクトのサーバアプリケーションのメニューと結合するための方法及びシステムを提供することである。

これら及び他の目的は、本発明の以下の説明から明らかとなるように、コンテナオブジェクト内に収容されたコンテナ（被収容物）オブジェクトと対話するためのコンピュータ方法及びシステムにより達成される。好ましい実施例において、コンテナオブジェクトは、そのコンテナオブジェクトと対話するため

のコンテナリソースを有するコンテナウインドウ環境を伴うコンテナアプリケーションを備えている。コンテナオブジェクトは、そのコンテナオブジェクトと対話するためのサーバリソースをもつサーバウインドウ環境を伴うサーバアプリケーションを備えている。本発明の方法は、コンテナウインドウ環境をディスプレイ装置に表示する。次いで、ユーザは、コンテナオブジェクトを選択する。コンテナオブジェクトの選択に応答して、この方法は、複数のサーバリソースを、表示されたコンテナウインドウ環境で積分する。ユーザがサーバリソースを選択するときには、この方法は、サーバリソース選択を処理するためにサーバアプリケーションを呼び出す。これに対し、ユーザがコンテナリソースを選択するときには、この方法は、コンテナリソース選択を処理するためにコンテナアプリケーションを呼び出す。

図面の簡単な説明

図1は、複合文書の一例を示す図である。

図2は、スケジュールデータ、予算データ及び説明データを複合文書にいかにより組み込むかを示す図である。

図3は、図1に示すサンプル複合文書であって、イン・プレーズ対話を行う前にワードプロセスアプリケーション内で編集されたときに現れる複合文書を示した図である。

図4は、複合文書内の位置でアクチベーションされたときに現れる埋め込まれるスプレッドシートオブジェクトを示す図である。

図5は、オブジェクトハンドラーとコンテナ及びサーバプロセスとの間の関係を示す図である。

図6は、リンクされるか又は埋め込まれるオブジェクトのサンプルインスタンスを示すブロック図である。

図7は、オブジェクトのパブリックビューを示すブロック図である。

図8は、オブジェクトに対して使用できるアクションを表示及び選択するためにコンテナアプリケーションにより与えられるサンプルユーザメニューを示す図である。

図9は、サーバアプリケーションメニューと図4に示す例のコンテナアプリ

ケーションメニューとの合体により得られる複合メニューバーを示す図である。

図10は、本発明の好ましい実施例において複合メニューバーを構成するメニューグループを示す図である。

図11は、典型的なシングル・ドキュメント・インターフェイスアプリケーションの成分ウインドウを示す図である。

図12は、マルチプル・ドキュメント・インターフェイスアプリケーションの成分ウインドウを示す図である。

図13は、埋め込まれたオブジェクトをその位置で編集するときのコンテナアプリケーションの典型的なウインドウハイアラキ構成を示すブロック図である。

図14は、事象駆動のウインドウオペレーティングシステム環境におけるメッセージ処理を示すフローチャートである。

図14Bは、イン・プレース対話をサポートするのに必要なパブリックインターフェイスを示すブロック図である。

図15は、`IOLEInPlaceFrame::SetMenu`方法の具現化を示すフローチャートである。

図16は、`IOLEInPlaceFrame::EnableModelElements`方法の具現化を示すフローチャートである。

図17は、`IOLEInPlaceParent::OnInPlaceActivate`方法の具現化を示すフローチャートである。

図18は、`IOLEInPlaceParent::OnUIActivate`方法の具現化を示すフローチャートである。

図19は、`IOLEInPlaceParent::OnUIDeactivate`方法の具現化を示すフローチャートである。

図20は、`IOLEInPlaceObject::InPlaceDeactivate`方法の具現化を示すフローチャートである。

図21は、`IOLEInPlaceObject::InPlaceUIDeactivate`方法の具現化を示すフローチャートである。

図22は、`IOLEInPlaceObject::Activate`方法の

具現化を示すフローチャートである。

図23は、ActivateUIファンクションの具現化を示すフローチャートである。

図24は、CreateObjectToolbarsファンクションの具現化を示すフローチャートである。

図25は、図4に示した例に対応する共用メニューデータ構造体のブロック図である。

図26は、ObjectSetMenuファンクションの具現化を示すフローチャートである。

図27は、Process_Object_Activationファンクションの具現化を示すフローチャートである。

図28は、APIファンクションObjectLoadをリンクしそして埋め込むオブジェクトの具現化を示すフローチャートである。

図29は、IOLEObject::DoVerb方法の具現化を示すフローチャートである。この方法は、コンテナオブジェクトと対話する一次方法である。

図30は、アクチベーション及びデアクチベーションメッセージを処理するためにMDI文書ウインドウのウインドウ手順によって呼び出されたファンクションProcess_Activation_Messageの具現化を示すフローチャートである。

図31は、Process_Mouse_LButtonUpファンクションの具現化を示すフローチャートである。

好ましい実施例の詳細な説明

目次

1. 概要
2. イン・プレース対話の概要
3. イン・プレース対話のウインドウサポート
4. イン・プレース対話API
 - 4.1 IOLEWindowインターフェイス

- 4.1.1 IOLEWindow::GetWindow
- 4. 2 IOLEInPlaceUIWindowインターフェイス
 - 4.2.1 IOLEInPlaceUIWindow::GetBorder
 - 4.2.2 IOLEInPlaceUIWindow::QueryBorderSpace
 - 4.2.3 IOLEInPlaceUIWindow::SetBorderSpace
- 4. 3 IOLEInPlaceFrameインターフェイス
 - 4.3.1 IOLEInPlaceFrame::SetMenu
 - 4.3.2 IOLEInPlaceFrames::InsertMenus
 - 4.3.3 IOLEInPlaceFrame::RemoveMenus
 - 4.3.4 IOLEInPlaceFrame::SetStatusText
 - 4.3.5 IOLEInPlaceFrame::EnableModeless
 - 4.3.6 IOLEInPlaceFrame::TranslateAccelerator
- 4. 4 IOLEInPlaceParentインターフェイス
 - 4.4.1 IOLEInPlaceParent::CanInPlaceDeactivate
 - 4.4.2 IOLEInPlaceParent::OnInPlaceActivate
 - 4.4.3 IOLEInPlaceParent::OnUIActivate
 - 4.4.4 IOLEInPlaceParent::OnUIDeactivate
 - 4.4.5 IOLEInPlaceParent::OnDeactivate
 - 4.4.6 IOLEInPlaceParent::ShadeBorder
 - 4.4.7 IOLEInPlaceParent::GetWindowContext
- 4. 5 IOLEInPlaceObjectインターフェイス
 - 4.5.1 IOLEInPlacobject::InPlaceDeactivate
 - 4.5.2 IOLEInPlacobject::InPlaceUIDeactivate
 - 4.5.3 IOLEInPlacobject::TranslateAccelerator
 - 4.5.4 IOLEInPlacobject::Activate
 - 4.5.5 IOLEInPlacobject::ResizeBorder
 - 4.5.6 IOLEInPlacobject::EnableModeless
 - 4.5.7 IOLEInPlacobject::SetVisRecr
- 4. 6 その他のサーバアプリケーションファンクション

4.6.1 ActivateUI

4.6.2 CreateNewMenu

4.6.3 CreateObjectToolbars

4.6.4 RemoveMenus

4. 7 APIヘルパーファンクションをリンクし埋め込むオブジェクト

4.7.1 SetActiveObjectHwnd

4.7.2 GetActiveObjectHwnd

4.7.3 ObjectCreateSharedMenu

4.7.4 ObjectDestroySharedMenu

4.7.5 ObjectShade

4.7.6 ObjectSetMenu

5. イン・プレース対話APIの使用

5. 1 その位置でのアクチベーションの手順

5.1.1 マルチプルドキュメントインターフェイスアプリケーション
内の位置でのアクチベーション

5. 2 プルダウンメニューメッセージ取り扱いのユーザ選択

5. 3 その位置でのデアクチベーションの手順

5. 4 コンテナアプリケーションを閉じる

5. 5 モードレスダイアログとの対話

5. 6 アクセラレータキーコンビネーションの取り扱い

1. 概要

本発明は、複合文書のコンテキストにおける埋め込まれたデータ又はリンクされたデータと対話するための「イン・プレース対話 (in-place interaction)」と称する一般的な方法を提供する。即ち、埋め込まれるか又はリンクされたデータと対話するのに使用されるアプリケーションは、複合文書を具現化するアプリケーションのウインドウコンテキスト (メニュー及びウインドウ) を介してユーザにアクセスすることができる。このアクセス性は、「その位置でのアクチベーション (activation in place)」と称する。好ましい実施例において、埋め込まれるか又はリンクされた (収容された) データがその位置でアクチベートされたと

きには、その収容されたデータを具現化するアプリケーションのメニューが、複合文書を具現化するアプリケーションのメニューと合体され、複合メニューバーを形成する。複合メニューバーにおけるメニューの順序は、1組のメニューグループによって決定される。各アプリケーションは、そのメニューをこれらのメニューグループに分類し、そのメニューグループの順序で複合メニューバーにメニューを配置する。次いで、複合メニューバーは、複合文書を具現化するアプリケーションのメニューバーとしてインストールされ、そしてこのアプリケーションのウインドウに送られたフィルタメッセージにメッセージハンドラーがインストールされる。ユーザがメニュー項目を選択するときには、メッセージハンドラーは、そのメニュー項目が、収容されたデータを具現化するアプリケーションのメニューに属するか、又は複合文書を具現化するアプリケーションのメニューに属するかを判断する。次いで、メッセージハンドラーは、その選択されたメッセージ項目に対応する入力メッセージを正しいアプリケーションに送信する。

本発明は、収容されたデータをその位置でアクチベートする1組の抽象クラス(インターフェイス)及びファンクションを定義する。(C++プログラミング言語において、抽象クラスとは、そのデータ及び方法の定義はもつが、これら方法に対する具現化をもたないクラスである。クラスインスタンスデータを操作するのに使用できる方法に対する実際のコードを与えるのは、クラスを具現化するアプリケーションの役目である。)複合文書を具現化するアプリケーションは、これらインターフェイスの幾つかを具現化する役目を果たし、そして収容されたデータを具現化するアプリケーションは、他のものを具現化する役目を果たす。

本発明の好ましい実施例において、複合文書を形成するアプリケーションプログラムは、別のアプリケーションにより発生されたリンクされるか又は埋め込まれるデータの操作を制御する。オブジェクト指向の用語においては、このデータをオブジェクトと称する。(参照文献B u d d, T. 「オブジェクト指向のプログラミングの紹介 (An Introduction to Object-Oriented Programming)」アジソン・ウェスリー・パブリッシング社、1991年は、オブジェクト指向の概念及び用語を紹介している。)複合文書にリンクされるか又は埋め込まれるオブジェクトは、文書内に「収容」される。又、複合文書は「コンテナ」オブジェ

クト

と称し、そして複合文書内に収容されるオブジェクトは「収容」又は「コンテニー」オブジェクトと称する。図1及び2を参照すれば、スケジュールデータ102及び予算データ103はコンテニーオブジェクトであり、そして複合文書101はコンテナオブジェクトである。ユーザは、予算データ103のようなコンテニーオブジェクトをユーザが編集したい旨をワードプロセッサに指示することができる。予算データ103を編集すべきことをユーザが指示すると、ワードプロセスプログラムは、その予算データを編集するのにどのアプリケーションを使用すべきか（例えば、スプレッドシートプログラム）を決定し、そしてそのアプリケーションに着手（スタートアップ）する。次いで、ユーザは、その着手されたアプリケーションを用いて予算データを操作することができ、その変更が複合文書に反映される。予算データが埋め込まれるオブジェクトとして記憶されるかリンクされるオブジェクトとして記憶されるかにかかわらず、同じ手順が使用される。

予算データを編集するのに用いられるアプリケーションがイン・プレース対話をサポートする場合には、それがワードプロセスプログラムによって着手されたときに、ワードプロセスプログラムのウインドウ環境内でアクチベートされる。図3及び4は、埋め込まれる予算データをその位置でアクチベートするプロセスを示している。

図3は、図1に示すサンプル複合文書であって、イン・プレース対話が行われる前にワードプロセスアプリケーション内で編集されたときに現れる複合文書を示す図である。コンテナアプリケーション301のメインウインドウは、タイトルバー302、メニューバー303及びクライアントウインドウ304を含んでいる。クライアントウインドウ304は、図1について述べた製造プロジェクトレポートを表示する。複合文書は、埋め込まれたスプレッドシートオブジェクト（予算データ305）を含む。ユーザが複合文書のネイティブテキストデータを編集するときには、メニューバー303は、図示したように現れる。即ち、これは、ワードプロセスアプリケーションと対話するに必要な全てのコマンドを含

む。

ユーザは、予算データ305を編集することを決定すると、スプレッドシートオブジェクト305を選択し、そしてオブジェクトを編集するようにワードプロセッサアプリケーションに要求する（例えば、マウスを用いたダブルクリックにより）。ワードプロセッサアプリケーションは、次いで、スプレッドシートアプリケーションに着手して、スプレッドシートオブジェクト305を編集するよう要求する。スプレッドシートアプリケーションは、ワードプロセッサアプリケーションのウィンドウ301及び304とメニューバー303とを用いてスプレッドシートオブジェクト305を編集するようにワードプロセッサアプリケーションと交渉する。

図4は、複合文書内の位置においてアクチベートされたときに現れる埋め込まれたスプレッドシートオブジェクトを示す図である。スプレッドシートオブジェクト405は、ワードプロセッサアプリケーションのクライアントウィンドウ404において直接編集される。タイトルバー402は、複合文書を具現化するアプリケーション、この場合はワードプロセッサアプリケーションが、複合文書「VAC1.DOC」内のスプレッドシートワークシートを編集することを反映するように変更される。又、メニューバー403は、ワードプロセッサアプリケーションからのメニューと、スプレッドシートアプリケーションからのメニューとを含む新たな複合メニューバーに変更される。更に、埋め込まれたスプレッドシートオブジェクト405の種々の特徴は、それがコンテナ複合文書内で編集されていることを反映するように変更される。陰影付けされた境界パターンの形態である選択ハイライト406がオブジェクトの周りに配置される。又、スプレッドシートアプリケーションの標準ツール、この場合は行列マーク407がスプレッドシートオブジェクトの周りに配置される。又、現在選択されたセルの周りにはスプレッドシート選択カーソル408が配置される。この点において、ユーザは、全てのスプレッドシートアプリケーションコマンドを用いてスプレッドシートオブジェクト405を編集する準備ができる。

好ましい実施例において、アプリケーションプログラム（アプリケーション）

は、オブジェクトをリンクし及び埋め込む構成体を用いて互いに共働し、複合文書の形成及び操作を行う。複合文書を形成するアプリケーションは、コンテナ（又はクライアント）アプリケーションと称し、そしてコンテニーオブジェクト

を形成及び操作するアプリケーションは、サーバアプリケーションと称する。アプリケーションは、コンテナ及びサーバの両方として働くことができる。即ちアプリケーションはオブジェクトを含むことができ、そしてアプリケーションが具現化するオブジェクトは他のオブジェクト内に含むことができる。図2を参照すれば、オブジェクトマネジメントプログラム201及びスプレッドシートプログラム204はサーバアプリケーションであり、そしてワードプロセスプログラム206はコンテナアプリケーションである。コンテナアプリケーションは、コンテナオブジェクト内で種々のオブジェクトを選択しそしてコンテニーオブジェクトを操作するための適切なサーバアプリケーションを呼び出すという役目を果たす。サーバアプリケーションは、コンテニーオブジェクトの内容を操作する役目を果たす。

好ましい実施例において、アプリケーションには、具現化とは独立したアプリケーションプログラミングインターフェイス（API）が与えられ、これは、オブジェクトをリンクしそして埋め込む機能を果たすものである。APIは、コンテナ及びサーバアプリケーションによって呼び出される1組のファンクションである。これらのファンクションは、他のものの中でも、コンテナアプリケーションがサーバアプリケーションとの間でメッセージ及びデータをやり取りするのに必要な設定及び初期化を管理する。

サーバアプリケーションの呼び出しは、サーバアプリケーションがコンテナアプリケーションとは個別のプロセスとして実行されるときには比較的低速である。ある状態においては、この低速さは特に望ましくない。例えば、ユーザが、多数のコンテニーオブジェクトを含む複合文書をプリントしようとする場合は、各コンテニーオブジェクトに対するサーバプロセスを呼び出してオブジェクトをプリントするよう各サーバプロセスに要求するのに、受け入れられないほど長い時間がかかる。この受け入れられない性能を改善するために、サーバアプリケー

ションは、ランタイム中に、コンテナアプリケーションのプロセスに動的にリンクして更に効率的なある機能を発揮できるようにするコードを与えることができる。このコードは「オブジェクトハンドラー」と称する。このオブジェクトハンドラーは、オブジェクトをリンクし埋め込むAPIがサーバプロセスをスター

トしてサーバプロセスにメッセージを通すのを回避できるようにサーバアプリケーションに代わる機能を発揮する。上記の例では、オブジェクトハンドラーは、オブジェクトをリンクし埋め込むAPIがコンテナオブジェクトのプリントを呼び出し得るようなプリントファンクションを与えることができる。

図5は、オブジェクトハンドラーとコンテナ及びサーバプロセスとの間の関係を示す図である。オブジェクトハンドラー502は、ランタイム中に、オブジェクトリンク及び埋め込みAPI503によってコンテナプロセスアドレススペース501にリンクされる。典型的に、オブジェクトリンク及び埋め込みAPI503は、オブジェクトハンドラー502を直接呼び出し、そしてコンテナアプリケーションコードは、ハンドラーがサーバプロセス507以外の機能を与えることに気付く必要はない。

1組のファンクションを与えるのに加えて、オブジェクトリンク及び埋め込み(OLE) APIは、コンテナアプリケーションがそれらの収容されたオブジェクトと通信できるようにする「インターフェイス」を定義する。インターフェイスは、ある入力、出力及び振る舞いルールにより遵守する1組の方法(C++言語での)である。収容されたオブジェクトが特定のインターフェイスをサポートする場合には、コンテナアプリケーションは、そのインターフェイスの方法を呼び出して、定義された振る舞いを行うことができる。特定の実施例では、コンテナアプリケーションは、オブジェクトデータを直接アクセスしない。むしろ、サポートされたインターフェイスを用いてオブジェクトデータをアクセスするのが好ましい。コンテナアプリケーションは、インターフェイスに対するポインタを介して収容されたオブジェクトへ結合される。コンテナアプリケーションは、インターフェイスの方法を呼び出すことによりオブジェクトデータをアクセスする。オブジェクトデータをアクセスするために、これら方法は、指定の

アクセスを要求するサーバアプリケーションへメッセージを送信する。好ましい実施例において、メッセージは、サーバアプリケーションがその基礎となるオペレーティングシステムにより与えられるプロセス間通信メカニズムを用いて個別のプロセスとして具現化されるときにコンテナアプリケーションとサーバアプリケーションとの間に送られる。

図6は、リンクされるか又は埋め込まれたオブジェクトのサンプルインスタンスを示すブロック図である。好ましい実施例において、インスタンスのレイアウトは、参考としてここに取り上げる「オブジェクト指向のプログラミング言語のためのコンパイラにおいて仮想ファンクション及び仮想ベースを具現化するための方法 (A Method for Implementing Virtual Functions and Virtual Bases in a Compiler for an Object Oriented Programming Language)」と題する米国特許出願第07/682,537号に規定されたモデルに合致する。このインスタンスは、各々のサポートされるインターフェイスに対するオブジェクトデータ構造601及びインターフェイスデータ構造61を含んでいる。オブジェクトデータ構造601は、インターフェイスデータ構造613に対するポインタ602を含み、そしてインスタンスのプライベートデータを含んでもよい。このサンプルインスタンスのプライベートデータは、クラス識別子603と、オブジェクトの記憶に対するハンドル604と、オブジェクトの状態を追跡するためのデータ605とを備えている。クラス識別子 (CLASS_ID) は、オブジェクトに対する適切なサーバアプリケーションをアクセスするのに使用される。これは、プログラミング言語に使用されるデータ構造「タイプ」と同様である。インターフェイスは、CLASS_IDを永続的なグローバルな登録簿へのインデックスとして用いることによりオブジェクトに対するサーバアプリケーションを決定することができる。永続的なグローバルな登録簿は以下で詳細に説明する。図6に示すように、各インターフェイスデータ構造613は、プライベートデータ構造606及び仮想ファンクションテーブル608を含む。プライベートデータ構造606は、仮想ファンクションテーブル608に対するポインタ607を含む。仮想ファンクションテーブル608は、インターフェイスの方法を具現化するコー

ド610、612に対するポインタ609、611を含む。

テーブル1

```
# define interface class
interface intf {public:
    virtual RETCODE fnc1(arg1, arg2)=0;
    virtual RETCODE fnc2(arg1, arg2)=0;

    virtual RETCODE fnc3()=0;

    ...

} :
```

テーブル1は、オブジェクトデータ構造601における第1エントリー `p i n t f`、に対するインターフェイスの定義を表している。テーブル1において、語「インターフェイス」は、C++クラスを意味するものと定義される。この定義は、3つの方法をそれらのパラメータと共に示している。各パラメータリストの端にある「=0」は、その方法がコードの具現化をもたないことを意味する。C++プログラミング言語において、これらのファンクションは「純粋な仮想ファンクション」と称される。純粋な仮想ファンクションをもつクラスは、抽象クラスと称される。

図7は、オブジェクトのパブリックビューを示すブロック図である。オブジェクトのパブリックビューは、オブジェクトが702ないし706をサポートする種々のインターフェイスである。各インターフェイスは、コンテナアプリケーションがオブジェクトをアクセスできるようにする方法を与える。各オブジェクトは、IUnknownインターフェイス702をサポートする。コンテナアプリケーションは、IUnknownインターフェイス702を使用して、オブジェクトが他のインターフェイスのどれをサポートするかを決定する。特定のオブジェクトに対するIUnknownインターフェイス702の具現化は、オブジェクトが他のどんなインターフェイスをサポートするかを知り、そしてそれらインターフェイスに対する呼び出しアプリケーションポインタへ復帰する。好ましい実施例では、方法 `IUnknown::QueryInterface` がこの

目的に使用される。インターフェイス703ないし706は、オブジェクトによってサポートすることのできる典型的なインターフェイスの例である。これらのインターフェイスは、IUnknownインターフェイスから導出される。例えば、IDataObjectインターフェイス703は、データを記憶すると共にオブジェクトからデータを検索するための方法を与える。IOLEContainer704は、オブジェクト内に収容されるコンテナーオブジェクトをリストするための方法を与える。IPersistStorageインターフェイス

705は、オブジェクトを永続的な記憶装置に記憶したりそこからオブジェクトを検索したりする方法を与える。IOLEObjectインターフェイス706は、コンテナーアプリケーションがユーザ選択アクションに対応するオブジェクトの機能を呼び出すところの方法を与える。

APIに加えて、本発明のオブジェクトリンク及び埋め込み構成体は、永続的なグローバルな「登録簿」によりコンテナー及びサーバアプリケーションへ情報を与える。この登録簿は、(1) オブジェクトの各タイプごとに、オブジェクトタイプを具現化するサーバアプリケーション、(2) 各サーバアプリケーションがコンテナーアプリケーションに与えるアクション、(3) 各サーバアプリケーションに対する実行可能なファイルが置かれた場所、及び(4) 各サーバアプリケーションがそれに関連したオブジェクトハンドラーを有するかどうか、といった情報のデータベースである。

2. イン・プレース対話の概要

オブジェクトがいったん文書にリンクされるか又は埋め込まれると、ユーザはオブジェクトを選択しそしてその選択されたオブジェクトに基づいてあるアクションを実行することを要求する。ユーザは、先ず、オブジェクトを選択しそしてオブジェクトに基づいて実行されるべきアクション（例えば、メニュー項目）を選択することによりアクションを要求する。次いで、具現化するサーバアプリケーションが呼び出され、その選択されたアクションを実行する。考えられるアクションの選択をユーザに表示しそしてユーザがアクションを選択できるようにする方法は多数あることが当業者に明らかであろう。好ましい実施例では、コンテ

ナーアプリケーションは、グローバルな登録簿から、その選択されたオブジェクトを具現化するサーバアプリケーションによってどんなアクションがサポートされるかを判断し、そしてアクションをメニューに表示する。

図8は、オブジェクトに対して使用できるアクションを表示しそして選択するためにコンテナアプリケーションによって与えられる例示的なユーザメニューを示す。メニュー項目803は、コンテナアプリケーションEdit（編集）メニュー802におけるオブジェクトのエントリーである。このエントリーは、現在選択されたオブジェクトに基づいて変化する。埋め込まれるか又はリンクさ

れたオブジェクトが選択されないときは、メニュー項目803は表示されない。サブメニュー804は、「エクセル・ワークシート・オブジェクト (Excel Worksheet Object)」によってサポートされるアクションを表示する。この例では、サポートされるアクションは、「Edit UN」「Open（オープン）」及び「Type（タイプ）」である。サブメニューの最初のアクション（例えば、「Edit」）は、ユーザがオブジェクトのマウスポインタ装置でダブルクリックするか又は機能的に等価なキーを入力するときに実行されるデフォルトアクションである。

ユーザが所望のアクションを選択すると（メニューから又はオブジェクトにおけるダブルクリック動作により）、コンテナアプリケーションは、サーバアプリケーションを呼び出し、コンテナアプリケーションに代わってどんなアクションを実行すべきかをそこに通す。コンテナアプリケーションは、オブジェクトに対するIOLEObjectインターフェイスを得そしてオブジェクトのDoVerb方法呼び出して、選択されたアクションをそこに通すことにより、これを行う。（DoVerb方法は、オブジェクトにおいてオブジェクトに特定のアクションを実行する。）サーバアプリケーションは、次いで、コンテナアプリケーションのコンテキスト内の位置でオブジェクトをアクチベートできるかどうかを決定する。もしそうならば、サーバアプリケーション及びコンテナアプリケーションは、それらのメニューを複合メニューバーに合体し、サーバアプリケーションツールバー、パレット、公式等の配置を交渉し、そして合体されたメッ

セージの取り扱いを設定する。この点において、サーバアプリケーションは、ユーザ入力を受け取る準備ができる。

図4の例を続けると、図示されたように、ユーザは、ワードプロセッサアプリケーションのウインドウ環境内の位置においてスプレッドシートオブジェクト（予算データ405）を編集する。図9は、サーバアプリケーションのメニューと、図4に示す例のコンテナアプリケーションメニューとの合体により生じる複合メニューバーを示す図である。複合メニューバー901は、ワードプロセッサアプリケーションからのメニュー902、905と、スプレッドシートアプリケーションからのメニュー903、904、906とを備えている。ユーザがこれらメ

ニューの1つから特定のメニュー項目を選択すると、コンテナアプリケーションは、合体されたメッセージハンドラーにより、メッセージをワードプロセッサアプリケーションへ發送すべきか、スプレッドシートアプリケーションへ發送すべきかを決定する。

本発明の好ましい実施例では、複合メニューバーは、1組の所定の規定に基づいて形成される。複合メニューバーに含まれるべき各アプリケーションメニューは、メニューグループに指定される。次いで、メニューは、その指定されたメニューグループに基づいて複合メニューバーに挿入される。

図10は、本発明の好ましい実施例において複合メニューバーを構成するメニューグループを示す図である。複合メニューバー1003は、コンテナアプリケーションからのメニューグループ1001と、サーバアプリケーションからのメニューグループ1002とを備えている。コンテナアプリケーションのメニューグループ1001は、ファイルグループと、コンテナグループと、ウインドウグループとを含む。サーバアプリケーションのメニューグループ1002は、編集グループと、オブジェクトグループと、ヘルプグループとを含む。好ましい実施例において、コンテナ及びサーバアプリケーションのメニューは、参考としてここに取り上げる1992年マイクロソフト社の「ザ・ウインドウズ・インターフェイス：アプリケーションデザインガイド（The Windows Interface: An Application Design Guide）」に規定されたマイクロソフトアプリケーションユー

ザインターフェイススタイルガイドラインに基づいて最終的な複合メニューバーにインターリーブされる。特に、複合メニューバー1003において、グループは、次の順序、即ちファイル、編集、コンテナー、オブジェクト、ウインドウ及びヘルプの順に、左から右へ配列される。

3. イン・ブレース対話のウインドウサポート

好ましい実施例においては、イン・ブレース対話APIは、その基礎となるウインドウシステム的能力を用いて具現化される。本発明は、その基礎となるウインドウシステムがマイクロソフトウインドウズ3.1オペレーティングシステム("Windows")に類似したものであると仮定して説明するが、本発明は、異なる基礎的なウインドウシステムでも実施できることが当業者に明らかであろう。マイ

クロソフトウインドウズ3.1オペレーティングシステムは、参考としてここに取り上げる1992年マイクロソフト社の「プログラマの参考書、第2巻：ファンクション (Programmer's Reference, Volume 2: Functions)」；1992年マイクロソフト社の「プログラマの参考書、第3巻：メッセージ、構造及びマクロ (Programmer's Reference, Volume 3: Messages, Structures, and Macros)」；及び1992年マイクロソフト社の「プログラミングの手引き (Guide to Programming)」に掲載されている。

ウインドウ環境では、アプリケーションは、単一文書インターフェイス又は多文書インターフェイスをサポートする。単一文書インターフェイス(SDI)アプリケーションは、一度に1つの文書(ファイル)と対話する。例えば、SDIをサポートするワードプロセスアプリケーションは、現在編集されているファイルをその一次ウインドウに表示する。多文書インターフェイス(MDI)のアプリケーションは、各文書に対して少なくとも1つのウインドウを割り当てることにより多数の文書(ファイル)と対話する。例えば、MDIをサポートするワードプロセスアプリケーションは、現在編集されている各ファイルを個別の文書ウインドウに表示する。ユーザは、所望の文書ウインドウのタイトルバーにおいてクリックするか、又はアプリケーションのウインドウメニュー上のリストからウ

インドウタイトルを選択することにより、ユーザが編集しようとするファイルの文書ウインドウを選択する。

図11は、典型的な単一文書インターフェイスアプリケーションの成分ウインドウを示す図である。典型的なSDIアプリケーションは、フレームウインドウ1101を与え、そしてアプリケーションによっては、埋め込まれたオブジェクトの存在に対し区画ウインドウ1105及び1106と、親ウインドウ1107とを更に与えることができる。SDIアプリケーションの場合には、フレームウインドウ1101は文書ウインドウでもある。区画ウインドウ1105、1106は、複合文書の多数のビューを与える。親ウインドウ1107は、オブジェクトが複合文書に最初に挿入されたときにオブジェクトの輪郭を描くようにコンテナアプリケーションによって形成される。図11に示す例では、埋め込まれたオブジェクトはスプレッドシートオブジェクトであり、これは、コンテナアプ

리케이션の親ウインドウ1107内に収容されたオブジェクトウインドウ1108内に表示される。オブジェクトウインドウ1108は、サーバアプリケーションによって所有される。フレームウインドウ1101は、タイトルバー1102と、メニューバー1103と、ツールバー1104とを含む。典型的に、ツールバー及びその他のアプリケーションに特定のツールは、フレームウインドウに取り付けられるか、又はコンテナアプリケーションの区画ウインドウに取り付けられる。又、これらは、図11に示すウインドウとは独立したウインドウであるフローティングパレットとしても現れ、従って、頂部に「浮動」するように見える。

図12は、典型的な多文書インターフェイスアプリケーションの成分ウインドウを示す図である。典型的なMDIアプリケーションは、ユーザが同じコンテナアプリケーション内から多数の成分文書を編集できるようにする。図12に示す例では、ユーザは、2つの文書ウインドウ1205、1206において2つの個別の成分文書を編集する。各文書ウインドウは、SDIアプリケーションと同様に、区画ウインドウを含むことができる。文書ウインドウ1205は、2つの区画ウインドウ1207、1208を含む。又、MDIアプリケーションは、S

DIアプリケーションと同様に、埋め込まれたオブジェクトを収容するための親ウィンドウ1209を与えることができる。図12は、オブジェクトウィンドウ1210内に存在する埋め込まれたスプレッドシートオブジェクトを示す。SDIアプリケーションの場合と同様に、アプリケーションに特定のツールがどこかに現れる。

SDI又はMDIのいずれかのアプリケーションにより管理されるウィンドウは、ハイアラキー形態で形成されそして維持される。図13は、埋め込まれたオブジェクトをその位置において編集するときのコンテナアプリケーションの典型的なウィンドウハイアラキーを示すブロック図である。このウィンドウハイアラキーは、コンテナアプリケーションからのコンテナアプリケーションウィンドウ1301と、サーバアプリケーションからのサーバアプリケーションウィンドウ1307とを備えている。コンテナアプリケーション1302は、そのフレームウィンドウ1303を管理し、該ウィンドウは文書ウィンドウ1304

を含み、該ウィンドウは区画ウィンドウ1305を含み、そして該ウィンドウは親ウィンドウ1306を含む。オブジェクトがその位置においてアクチベートされるときは、サーバアプリケーション1308は、埋め込まれたオブジェクトに対するルートウィンドウ1309と、これが必要とする子ウィンドウとを形成する。オブジェクトルートウィンドウ1309は、オブジェクト子ウィンドウ1310、1311、1312を含む。

各アプリケーションは、個別のプロセスとして具現化されるときに、アプリケーションのウィンドウハイアラキーに存在するウィンドウに接続された事象を受け取るための入力待ち行列を含んでいる。図13のウィンドウハイアラキーは、2つの異なるアプリケーションによってサポートされる。従って、コンテナアプリケーションに属するウィンドウと、サーバアプリケーションに属するウィンドウとに関連した個別の入力待ち行列がある。入力待ち行列1313は、コンテナアプリケーションウィンドウ1301に関連している。入力待ち行列1314は、サーバアプリケーションウィンドウ1307に関連している。ユーザがマウスでクリックするか、又はこれらウィンドウの1つに対してキーストロークを

入力したときに、その基礎となるウインドウシステムは、コンテナー入力待ち行列1313又はサーバアプリケーション待ち行列1314のいずれかに適当なメッセージを入力する。

4. イン・ブレース対話API

オブジェクトリンク及び埋め込みAPIは、イン・ブレース対話をサポートするためにコンテナーアプリケーションとサーバアプリケーションが通信するところのファンクションを与え、そのインターフェイスを定義する。これらインターフェイスの方法及びその他のAPIファンクションは、ユーザ入力の通常の処理中にアプリケーションコードによって呼び出される。事象駆動されるウインドウシステムにおいては、アプリケーションは、ユーザが特定のメニュー項目又はオブジェクトを選択したことを指示するメッセージを受け取るのに応答して適当な方法又はファンクションを呼び出す。

図14は、事象駆動式ウインドウオペレーティングシステム環境におけるメッセージの処理を示すフローチャートである。各ウインドウは、そのウインドウが

形成されるときにその基礎となるウインドウシステムと整列されるそれ自身のメッセージハンドラーを有している。アプリケーション入力待ち行列（例えば、コンテナーアプリケーションの入力待ち行列1313）にメッセージが受け取られると、アプリケーションは、メッセージをフィルタし、変換し、又はウインドウシステムへ発送する。ウインドウシステムのディスパッチャーは、次いで、メッセージにおいて指示された特定のウインドウに対して既に整列されているメッセージ取扱ファンクション（メッセージハンドラー）にメッセージを送信する。メッセージを受け取ると、メッセージハンドラーは、メッセージを処理する。この処理は、オブジェクトリンク及び埋め込みAPIの使用を含む。ステップ1401及び1402は、メッセージポンプを形成する。ステップ1401において、アプリケーションは、その入力待ち行列におけるメッセージを待機する。ステップ1402において、アプリケーションは、メッセージを適宜フィルタ又は変換するか、又はメッセージをウインドウシステムのディスパッチファンクションへ発送する。ステップ1403及び1404は、ウインドウシステムのディスパッ

チファンクションにおいて、メッセージを適当なウインドウメッセージハンドラーへ発送するステップである。ステップ1403において、ウインドウシステムディスパッチャーは、メッセージにおいて指示されたウインドウに対しメッセージハンドラーを位置決めする。ステップ1404において、ウインドウシステムディスパッチャーは、その位置決めされたメッセージハンドラーへメッセージを送信する（例えば、メッセージハンドラーを呼び出すことにより）。

ステップ1405ないし1412は、ウインドウに対する典型的なメッセージハンドラーを構成する。メッセージハンドラーは、「ウインドウ手順」とも称する。好ましい実施例において、アプリケーションが特定のウインドウに対するウインドウ手順を与えない場合には、その基礎となるウインドウシステムは、DefaultWindowProcと称するデフォルトウインドウハンドラーを与える。ステップ1405ないし1408において、アプリケーションは、メッセージをデコードして、どのタイプの事象が生じたかを決定する。典型的に、各タイプの事象ごとに、アプリケーションは、ステップ1409ないし1412に示すように異なるファンクションを呼び出す。これらのファンクションは、次いで、オブ

ジェクトリンク及び埋め込みAPIを使用する。例えば、メニュー事象が受け取られたときには、アプリケーションは、ステップ1411において、メニュー事象を処理するファンクションを呼び出す。ステップ1411は、コンテニーオブジェクトをその位置においてアクチベートするProcess_ObjectActivationファンクション（ステップ1413として示す）を呼び出す。以下に詳細に述べるように、Process_Object_Activationファンクションは、オブジェクトリンク及び埋め込みAPIを用いて、コンテニーオブジェクトをアクチベートする。

イン・プレース対話APIは、次のインターフェイス、即ちIOLEWindow; IOLEInPlaceUIWindow; IOLEInPlaceFrame; IOLEInPlaceParent; 及びIOLEInPlaceObjectを定義する。IOLEWindowインターフェイスは、他のインターフェイスの1つに関連したウインドウハンドルを検索するための方法を与える

ものである。IOLEInPlaceUIWindowインターフェイスは、サーバアプリケーションがウインドウツールの配置に対してコンテナアプリケーションと交渉する方法を与える。IOLEInPlaceFrameインターフェイスは、サーバアプリケーションがコンテナアプリケーションのフレームウインドウと通信する方法を与える。IOLEInPlaceParentインターフェイスは、サーバアプリケーションがコンテナアプリケーションの親ウインドウと通信する方法を与える。IOLEInPlaceObjectインターフェイスは、コンテナアプリケーションがサーバアプリケーションをアクチベート及びデアクチベートする方法を与える。図14Bは、イン・ブレース対話をサポートするのに必要なパブリックインターフェイスを示すブロック図である。コンテナオブジェクト14B01は、IOLEWindowインターフェイス14B02、IOLEInPlaceParentインターフェイス14B03及びIOLEInPlaceFrameインターフェイス14B04をサポートする。コンテナオブジェクト14B05は、IOLEInPlaceObjectインターフェイス14B06をサポートする。これらインターフェイスの各々について、以下に詳細に説明する。

4.1 IOLEWindowインターフェイス

テーブル2は、IOLEWindowインターフェイスをリストしたものである。オブジェクト指向の用語において、IOLEWindowインターフェイスは、他のイン・ブレース対話インターフェイスの「基本的クラス」である。従って、他のインターフェイスは、IOLEWindowインターフェイスから導出され、そのパブリック方法を継承する。IOLEWindowインターフェイスには、GetWindowと称する1つのパブリック方法しかない。

テーブル2

```
interface IOLEWindow: public IUnknown {public:
    virtual SCODE GetWindow (HWND FAR*phwnd)=0;
}
```

4.1.1 IOLEWindow::GetWindow

GetWindow方法は、これが呼び出されたIOLEInPlaceUIWindowNIOLEInPlaceFrame、IOLEInPlaceParent又はIOLEInPlaceObjectインターフェイスに対応するウィンドウハンドル（独特のウィンドウ識別子）を検索する。検索されたウィンドウハンドルは、通常、その基礎となるウィンドウシステムファンクションを呼び出すときに使用される。

4. 2 IOLEInPlaceUIWindowインターフェイス

テーブル3は、IOLEInPlaceUIWindowインターフェイスをリストする。IOLEInPlaceUIWindowインターフェイスの方法は、コンテナアプリケーションの文書及び区画ウィンドウ内のツール配置を交渉するためにサーバアプリケーションによって呼び出される。

テーブル 3

```
interface IOLEInPlaceUIWindow: public IOLEWindow {public:
    virtual SCODE GetBorder(RECT borderRect)=0;
    virtual SCODE QueryBorderSpace(RECT widthRect)=0;
    virtual SCODE SetBorderSpace(RECT widthRect)=0;
}
```

4. 2. 1 IOLEInPlaceUIWindow::GetBorder

GetBorder方法は、サーバアプリケーションがそのツール（子ウィンドウとして実現される）を配置できるようにする位置を検索する。この方法は、文書インターフェイスが呼び出されるか区画インターフェイスが呼び出されるかに基づいて、文書又は区画ウィンドウのフレーム内に配置された四角形を返送する。サーバアプリケーションは、この四角形を検索すると、ツールを配置するのに必要なスペースの巾を決定することができると共に、QueryBorderSpace方法を用いてこのスペースを要求することができる。コンテナアプリケーションによって返送された四角形が拒絶される場合には、サーバアプリケーションは、その位置でのアクチベーションを続けないように選択することもできるし、或いはそのツールをアクチベートしないように選択することもできる。

4 2. 2 IOLEInPlaceUIWindow::QueryBorderSpace

`QueryBorderSpace`方法は、サーバアプリケーションがそのツールを配置することのできる区画又は文書ウィンドウ内の指定量のスペースを検索する。この方法は、サーバアプリケーションがそのツール配置に必要な`GetBorder`への以前の通話から検索された四角形内の1つのパラメータ、即ち四角形のボーダースペース、を取り出す。この方法は、文書又は区画ウィンドウが要求を受け入れられるかどうかの指示を返送する。要求を受け入れられない場合には、サーバアプリケーションは、異なる四角形でこの方法を再び呼び出し、その位置でのアクチベーションを続けないように選択し、又はそのツールをアクチベートしないように選択することができる。

4. 2. 3 `IOLEInPlaceUIWindow::SetBorderSpace`

`SetBorderSpace`方法は、それに関連したコンテナアプリケーションに、サーバアプリケーションのツールを配置するために区画ウィンドウ又は文書ウィンドウに指定のスペースを実際に割り当てていることを知らせる。この方法は、`QueryBorderSpace`への以前の通話において区画又は文書ウィンドウからスペースが首尾良く要求された後に呼び出される。サーバアプリケーションは、それが必要とするスペースを割り当てる役目を果たす。この方法は、サーバアプリケーションがその子ウィンドウに割り当てようとする1つの

パラメータ、即ち四角形のスペース、を取り出す。指定された四角形は、以前に首尾良く要求されたものより小さくてもよい。「指定された」という用語は、パスしたパラメータを指し、そして「特定された」という用語は、特定の方法が属するインターフェイス、クラス、ウィンドウ又はオブジェクトを指す。この方法は、特定された区画又は文書ウィンドウのユーザインターフェイスリソースのいずれかを必要に応じて移動し又はサイズ決めする。

4. 3 `IOLEInPlaceFrame`インターフェイス

テーブル4は、`IOLEInPlaceFrame`インターフェイスをリストする。この`IOLEInPlaceFrame`インターフェイスは、そのコンテナアプリケーションのフレームウィンドウと通信するためにサーバアプリケーションにより呼び出される方法を与える。

テーブル 4

```

interface IOLEInPlaceFrame: public IOLEInPlaceUIWindow {public:
    virtual SCODE SetMenu(HANDLE hSharedMenu, HWND hwndObject)=0;
    virtual SCODE InsertMenu(HANDLE hmenu, UNIT FAR*lpiMenuCounts)=0;
    virtual SCODE RemoveMenu(HANDLE hmenu)=0;
    virtual SCODE SetStatusText(LPSTR lpszStatusText)=0;
    virtual SCODE EnableModeless(BOOL fEnable)=0;
    virtual SCODE TranslateAccelerator(LPMSG lpmsg, WORD WID)=0;
}

```

4. 3. 1 IOLEInPlaceFrame::SetMenu

SetMenu方法は、指定された複合メニューバーをコンテナアプリケーションのメニューバーとしてインストールしたり除去したりし、そして複合メニューバーに対するメッセージハンドラーをインストールする。図15は、IOLEInPlaceFrame::SetMenu方法の具現化を示すフローチャートである。この方法は、コンテナアプリケーションがMDIアプリケーションであるかSDIアプリケーションであるかに基づいて異なるメカニズムを使用して複合メニューバーをインストールする。ステップ1501において、この方法は、指定された複合メニューバーがNULLであるかどうかを決定し、もしそうであれば、ステップ15

02へ続き、さもなければ、ステップ1503へ続く。ステップ1502では、この方法は、ヘルパーファンクションObjectSetMenuDescriptorを呼び出して、複合メニューバーに対するメッセージハンドラーを除去しそして復帰する。ステップ1503では、この方法は、コンテナアプリケーションがSDIアプリケーションであるかどうかを決定し、もしそうであれば、ステップ1504へ続き、さもなければ、ステップ1505へ続く。ステップ1504においては、この方法は、その基礎となるウインドウシステムファンクションSetMenuを呼び出し、複合メニューバーをコンテナアプリケーションフレームウインドウのメニューバーとしてインストールし、そしてステップ1507へ続く。ステップ1505では、この方法は、フレームウインドウにメッセー

ジを送り、そのMDIメニュー設定を実行するように通知する。ステップ1506では、この方法は、その基礎となるウインドウシステムファンクションDrawMenuBarを呼び出してメニューバーを再描写する。ステップ1507では、この方法は、ヘルパーファンクションObjectSetMenuDescriptorを呼び出して、複合メニューバーに対するメッセージハンドラーをインストールする。ステップ1508では、この方法は、そのメニューバーを切り換えるときに必要となる他のプロセスを実行し、次いで、復帰となる。

4. 3. 2 IOleInPlaceFrame::InsertMenus

Code Table 1

```

VOID IOleInPlaceFrame::InsertMenus (hmenu, ContrCounts) {
1  if there are File Group menus present {
2      for each filegroupmenu {
3          hfilemenu = CreateMenu ( );
4          InsertMenu (hmenu, MF_APPEND, MF_BYPOSITION | MF_POPUP, hfilemenu);
5          ContrCounts[0] = ContrCounts[0] + 1;
6          for each filegroupmenu_item {
7              InsertMenu (hfilemenu, MF_APPEND, MF_BYPOSITION | MF_STRING,
8                  item_id, "string to be displayed");
9          }
10     };
11  if there are Container Group menus present {
12      for each containergroupmenu {
13          hcontmenu = CreateMenu ( );
14          InsertMenu (hmenu, MF_APPEND, MF_BYPOSITION | MF_POPUP, hcontmenu);
15          ContrCounts[1] = ContrCounts[1] + 1;
16          for each contgroupmenu_item {
17              InsertMenu (hcontmenu, MF_APPEND, MF_BYPOSITION | MF_STRING,
18                  item_id, "string to be displayed");
19          }
20     };
21  if there are Window Group menus present {
22      for each windowgroupmenu {
23          hwndmenu = CreateMenu ( );
24          InsertMenu (hmenu, MF_APPEND, MF_BYPOSITION | MF_POPUP, hwndmenu);
25          ContrCounts[2] = ContrCounts[2] + 1;
26          for each wndgroupmenu_item {
27              InsertMenu (hwndmenu, MF_APPEND, MF_BYPOSITION | MF_STRING,
28                  item_id, "string to be displayed");
29          }
30     };
31  return ( );
}

```

InsertMenus方法は、コンテナアプリケーションのメニューを、サブアプリケーションによって形成される複合メニューバーに挿入する。コード

テーブル1は、IOLEInPlaceFrame::InsertMenus方法を具現化するための擬似コ

ードを示す。この方法は、複合メニューバーと、メニューカウントのアレーとの2つのパラメータをとる。コンテナアプリケーションのメニューによって表されたメニューグループの各々に対し、そのグループに対するメニューを挿入するループがある。ライン1ないし10において、何らかのファイルグループメニューがある場合には、この方法は、これらのメニューを複合メニューバーに挿入し、そしてファイルグループに対応するインデックスにおいてメニューカウントアレーを増加する。(例えば、図4の例に示されたメニューバーが使用される場合には、インデックス=0である。)ライン11ないし20において、何らかのコンテナグループメニューが存在する場合には、この方法は、これらのメニューを複合メニューバーに挿入し、そしてコンテナグループに対応するインデックスにおいてメニューカウントアレーを増加する。最後に、ライン21ないし30において、追加されるべきウインドウグループメニューがある場合には、この方法は、これらメニューを複合メニューバーに挿入し、そしてウインドウグループに対応するインデックスにおいてメニューカウントアレーを増加する。この方法の完了時に、各インデックスにおいてメニューカウントアレーに記憶される値は、その特定のメニューグループに対してコンテナアプリケーションが挿入したメニューの数を指示する。この方法は、その基礎となるウインドウシステムから標準的なファンクション(CreateMenu及びInsertMenu)を呼び出し、コンテナアプリケーションに対するメニューを形成すると共に、それらを複合メニューバーに挿入する。

4. 3. 3 IOLEInPlaceFrame::RemoveMenus

RemoveMenus方法は、サーバアプリケーションが複合メニューバーの割り当てを解除する前に、コンテナアプリケーションがそのメニューを複合メニューバーから除去できるようにする。この方法は、IOLEInPlaceObject::InPlaceDeactive方法から呼び出される。RemoveMenus方法は、1つのパラメータ、即ちコンテナメニューが記憶される複合メニューバーのハンドルを取り出す。複合メニューバーは、この方法が呼び出される前に全てのサーバメニ

ユーをクリアすることが予想される。

4. 3. 4 IOLEInPlaceFrame::SetStatusText

`SetStatusText`方法は、サーバアプリケーションがコンテナアプリケーションのフレームウインドウの状態ウインドウ（もし1つあれば）をセットできるようにする。通常は、状態ウインドウは、フレームウインドウの底部に配置され、現在選択に対応する状態又はヒント情報を含んでいる。`SetStatusText`方法は、コンテナアプリケーションに特定のもので、コンテナアプリケーションがその状態ウインドウをセットするために通常行うどんなオペレーションも実行する。この方法は、1つのパラメータ、即ち状態ウインドウに挿入するためのテキストストリングを取り出す。

4. 3. 5 IOLEInPlaceFrame::EnableModeless

`EnableModeless`方法は、コンテナアプリケーションに対して現在表示されているモードレスダイアログをイネーブル又はディスエイブルするものである。モードレスダイアログとは、ユーザによって明確に閉じられるまで表示される入力ウインドウである。このウインドウが表示される間に、ユーザは他のウインドウと対話することができる。一方、モードダイアログは、ユーザが受け入れられる入力を入れるまで他のウインドウ処理を阻止する入力ウインドウである。この方法は、モードダイアログを表示したいが、それに関連したコンテナアプリケーションがモードレスダイアログを既に表示しているときにサーバアプリケーションによって呼び出される。

図16は、`IOLEInPlaceFrame::EnableModeless`方法を具現化するフローチャートである。この方法は、コンテナアプリケーションのモードレスダイアログを隠し、そして再び呼び出されたときに、モードレスダイアログを復帰させる。指定されたフラグ `fEnable` が真である場合には、その隠されたダイアログが表示され、さもなくば、現在表示されているいかなるモードレスダイアログも隠される（表示から除去されるが、内部メモリのデータ構造は割り当て解除されない）。ステップ1601において、この方法は、`fEnable` が真であるかどうか決定し、もしそうならば、ステップ1602へ続くが、さもなくば、ステッ

プ1603へ続く。ステップ1602において、この方法は、その基礎となるウインドウシステムファンクションShowWindowを呼び出し、既にセーブされたモードレスダイアログに関連したウインドウを回復させ、次いで、復帰す

る。ステップ1603において、この方法は、現在表示されている次のモードレスダイアログのウインドウハンドルをセーブする。ステップ1604において、この方法は、その基礎となるウインドウシステムファンクションShowWindowを呼び出し、モードレスダイアログに関連したウインドウを隠す。ステップ1605において、このファンクションは、表示されたモードレスダイアログが更にあるかどうかを調べるようにチェックを行い、もしあれば、このファンクションはステップ1603へ戻るようにループし、さもなければ、復帰する。

4. 3. 6 IOLEInPlaceFrame::TranslateAccelerator

TranslateAccelerator方法は、サーバアプリケーションがそれ自身確認しないキーストロークを受け取るときに、アクセラレータキー組合せを処理できるようにする。アクセラレータキー組合せは、メニューコマンドに対するキーボードショートカットであり、これについては以下で述べる。TranslateAccelerator方法は、サーバアプリケーションメッセージポンプにおいて呼び出されるファンクションObjectTranslateAcceleratorによって間接的に呼び出される。コンテナアプリケーションは、その通常のアクセラレータ処理を実行し、そしてアクセラレータが処理されたかどうかの指示を返送しなければならない。この値は、次いで、ファンクションObjectTranslateAcceleratorによってサーバアプリケーションへ通される。メッセージはサーバアプリケーションからコンテナアプリケーションへ転送されているので、その基礎となるウインドウシステムは、指定されたメッセージに関連した付加的なキー状態又はメッセージ情報を保持していない。

4. 4 IOLEInPlaceParentインターフェイス

テーブル5は、IOLEInPlaceParentインターフェイスをリストしている。このIOLEInPlaceParentインターフェイスは、親

ウインドウと通信するためにサーバアプリケーションによって呼び出された方法を与える。又、このウインドウは、オブジェクトに対するイン・プレース「コンテナサイト」とも称する。

テーブル 5

```
interface IOLEInPlaceParent: public IOLEWindow {public:
    virtual SCODE CanInPlaceActivate()=0;
    virtual SCODE OnInPlaceActivate()=0;
    virtual SCODE OnUIActivate()=0;
    virtual SCODE OnUIDeactivate()=0;
    virtual SCODE OnDeactivate()=0;
    virtual SCODE ShadeBorder(LPLECT lprect, DWORD grfStatus)=0;
    virtual SCODE GetWindowContext(IOLEInPlaceFrame* pFrame,
    IOLEInPlaceUIWindow* pDoc, IOLEInPlaceUIWindow* pPane, LPRECT lprectChild
    Position, HANDLE* hAccelTable)=0;
}
```

4. 4. 1 IOLEInPlaceParent::CanInPlaceActivate

CanInPlaceActivate 方法は、コンテナアプリケーションがイン・プレース対話をサポートするかどうかを判断するためにサーバアプリケーションによって使用される。この方法は、選択されたコンテナーオブジェクトに代わってアクチベーションを受け入れるか又は拒絶する機会をコンテナアプリケーションに与える。この方法は、コンテナアプリケーションがイン・プレース対話を許容するかどうかの指示を返送する。

4. 4. 2 IOLEInPlaceParent::OnInPlaceActivate

OnInPlaceActivate 方法は、サーバアプリケーションが（アクチベーション時に）新たな複合メニューバーを形成する前に必要なオペレーションを実行する機会をコンテナアプリケーションに与えるようにサーバアプリケーションによって呼び出される。図17は、IOLEInPlaceParent::OnInPlaceActivate 方法の具現化を示すフローチャートである。この具現化において行われるオペレーションは、コンテナーオブジェクトがアクチベートされたことを指示するフラ

グをステップ1701においてセットすることだけである。この情報は、特定のオブジェクトの親コンテナオブジェクトがアクチベート又はデアクチベートするように求められたときに後で使用される。このフラグは、親コンテナアプリケーションに、それ自身のユーザインターフェイスをアクチベート又はデアクチ

ベートするのではなく、その中に収容されたオブジェクト（ネスト状オブジェクト）をアクチベート又はデアクチベートする必要があるかどうかを知らせる。

4. 4. 3 IOLEInPlaceParent::OnUIActivate

OnUIActivate方法は、コンテナオブジェクトをその位置においてアクチベートする準備として全てのコンテナアプリケーションメニュー及びツールを除去する。図18は、IOLEInPlaceParent::OnUIActivate方法の具現化を示すフローチャートである。この方法によって行われるステップは、コンテナオブジェクトがその位置でアクチベートされたオブジェクトそれ自体であるかどうかに基づいている。ステップ1801において、この方法は、コンテナオブジェクトがその位置においてアクチベートされたかどうか判断する。もしそうでなければ、この方法は、ステップ1802へ続き、さもなければ、ステップ1803へ続く。ステップ1802において、コンテナオブジェクトは最上レベルのコンテナオブジェクトである（その位置でアクチベートされない）ので、この方法は、その通常の手順を用いて、コンテナアプリケーションメニュー及び余分なツールを除去し、そして復帰する。ステップ1803において、コンテナオブジェクトはコンテナオブジェクトでもあるので、この方法は、コンテナオブジェクトをコンテナオブジェクトとして処理する方法をアクセスするためにオブジェクト自身のIOLEInPlaceObjectを検索する。ステップ1804において、この方法は、コンテナオブジェクトのアクチベート方法呼び出してそれ自身をデアクチベートする。ステップ1805において、この方法は、全てのコンテナオブジェクトの文書及び区画ウインドウレベルツールを隠す。ステップ1806において、この方法は、親コンテナオブジェクトのシェードボーダー方法呼び出して、コンテナオブジェクトの周りからのイン・プレーズ対話ユーザフィードバックを除去し、そして復帰する。コンテナオ

プロジェクトのオブジェクトウィンドウは、実際には、後でデアクチベートされる（例えば、コンテナーオブジェクトがデアクチベートするときに）。

4. 4. 4 IOLEInPlaceParent::OnUIDeactivate

OnUIDeactivate方法は、そのユーザインターフェイスリソースのデアクチベート動作の終わりにサーバアプリケーションによって呼び出され、

その親コンテナアプリケーションがそれ自身のユーザインターフェイスをアクチベートするか又はその親コンテナアプリケーションを呼び出して、その親コンテナアプリケーションがそのユーザインターフェイスをアクチベートできるようにする。図19は、IOLEInPlaceParent::OnUIDeactivate方法の具現化を示すフローチャートである。この方法は、コンテナオブジェクトがコンテナーオブジェクトそれ自体であるか又は最上レベルのコンテナオブジェクトであるかに基づいて2つの異なる振る舞いを与える。前者の場合に、このコンテナがその位置でアクチベートされる新たなオブジェクトになった場合に、それ自身のユーザインターフェイスがアクチベートされ、さもなくば、コンテナは、その親コンテナアプリケーションに、そのユーザインターフェイスをアクチベートするように要求する。後者の場合には、コンテナアプリケーションは、通常の手順を用いてそのユーザインターフェイスを復帰させる。ステップ1901において、この方法は、コンテナアプリケーションがコンテナーオブジェクトをアクチベートしたことを指示するフラグをクリアする。ステップ1902において、この方法は、特定されたコンテナオブジェクトがコンテナーオブジェクトであるかどうかを決定し、もしそうでなければ、ステップ1903へ続き、さもなくば、ステップ1905へ続く。ステップ1903において、この方法は、通常の手順を用いてコンテナアプリケーションのメニュー及びそのタイトルバーをセットし、そしてステップ1904へと続いて、入力フォーカスを所望のウィンドウへセットし、復帰する。入力フォーカスは、特定のウィンドウがキーボード入力を受け取るべきときにその特定のウィンドウをセットする。ステップ1905において、この方法は、フラグABOUT_TO_ACTIVATEを検討してコンテナオブジェクトがアクチベートされたオブジェクトになりつつあるかを

判断し、もしそうでなければ、ステップ1906へ続き、さもなければ、ステップ1907へ続く。(ABOUT_TO_ACTIVATEフラグは、例えば、以下に詳細に述べるProcess_Mouse_LButtonUpファンクションにおいてコンテナアプリケーションがユーザによって選択されたときにセットされる。) ステップ1906において、この方法は、コンテナオブジェクトのIOLEInPlaceParent::OnUIDeactivate方法呼び出して、親コンテナオブジ

ェクトのコンテナアプリケーションのユーザインターフェイスをアクチベートし、そして復帰する。ステップ1907において、この方法は、ファンクションActivateUIを呼び出し、コンテナアプリケーションのユーザインターフェイスをアクチベートし、そして復帰する。

4. 4. 5 IOLEInPlaceParent::OnDeactivate

OnDeactivate方法は、サーバアプリケーションによって呼び出され、それに関連したコンテナアプリケーションに、アクチベートされたコンテナオブジェクトが完全にデアクチベートされる前にそのコンテナオブジェクトに関連したフラグをセットするか又はいずれかのリソースを解放する機会を与える。この方法は、コンテナオブジェクトのIOLEInPlaceObject::InPlaceDeactivate方法から呼び出される。

4. 4. 6 IOLEInPlaceParent::ShadeBorder

ShadeBorder方法は、選択されるか又は選択解除されようとしているコンテナオブジェクトの周りに陰影付けされたパターンボーダーを描くか又はそれを除去する。陰影付けされたパターンボーダーは、コンテナオブジェクトがその位置でアクチベートされたというユーザフィードバックを与えるのに用いられる。この方法は、ヘルパーオブジェクトリンク及び埋め込みAPIファンクションObjectShadeを呼び出して、適切なシェードパターンを形成することができる。この方法は、2つのパラメータ、即ちボーダーを配置しなければならないオブジェクトの周りの四角形と、1組のフラグとを取り出す。この1組のフラグは、ボーダーをオン(SHADEBORDER_ON=1)にすべきかオフにすべきかを指示すると共に、ボーダーをアクティブウィンドウのタイ

トルバーに含まれたテキストと同じカラーで描くべき (SHADEBORDER ACTIVE=1) か又はディスエイブルされたテキストと同じカラーで描くべきかを指示する。アクティブウインドウは、入力フォーカスをもつウインドウである。

4. 4. 7 IOLEInPlaceParent::GetWindowContext

GetWindowContext方法は、特定のコンテニーオブジェクトに関連した1組のコンテナアプリケーションインターフェイスを返送する。より

詳細には、次のパラメータを返送する。

pFrame: これはIOLEInPlaceフレームインターフェイスに対するポインタである。

pDoc: これはIOLEInPlaceUIWindowインターフェイスに対するポインタである。

pPane: これはIOLEInPlaceUIWindowインターフェイスに対するポインタである。

lprctChildPson: これは関連するIOLEInPlaceParentインスタンスが親ウインドウ内にオブジェクトのオブジェクトウインドウを表示する場所に対するポインタである。

hAccelTable: これはコンテナアプリケーションのアクセラレータテーブル (以下に述べる) に対するハンドルである。

これらの値は、アクチベーション及びデアクチベーションを交渉しそして取り扱うためにサーバアプリケーションによって使用される。この方法は、これらインターフェイスのインスタンスを形成してそれをコンテナアプリケーションの当該フレーム、文書、区画及び親ウインドウと関連させる。

4. 5 IOLEInPlaceObjectインターフェイス

テーブル6は、IOLEInPlaceObjectインターフェイスをリストしたものである。IOLEInPlaceObjectインターフェイス方法は、収容されたオブジェクトをアクチベート及びデアクチベートするためにコンテナアプリケーションによって呼び出される。これら方法の幾つかは、ネスト

状に収容されたオブジェクトを収容ハイアラキーによりアクセスする。他の方法は、編集メニューを表示するコンテニーオブジェクトである現在アクティブなオブジェクトのみをアクセスする。別の具現化は、このインターフェイスを2つの他のものに分割することであり、その1つは、アクティブなオブジェクトのみをアクセスすることであり、そしてもう1つは、収容ハイアラキーによりコンテニーオブジェクトをアクセスすることである。

テーブル 6

```
interface IOLEInPlaceObject: public IOLEWindow {public:
    virtual SCODE InPlaceDeactivate()=0;
    virtual SCODE InPlaceUIDeactivate()=0;
    virtual SCODE TranslareAccelerator(LPMSG lpmsg)=0;
    virtual SCODE Activate(BOOL fActivate, BOOL fDocActivate)=0;
    virtual SCODE ResizeBorder(RECT borderRect)=0;
    virtual SCODE EnableModeless(BOOL fEnable)=0;
    virtual SCODE SetVisRect(LPRECT lprect)=0;
}
```

4. 5. 1 IOLEInPlaceObject::InPlaceDeactivate

InPlaceDeactivate方法は、コンテナアプリケーションによって呼び出され、「undo」オペレーションがコンテニーオブジェクトをアクセスする必要がもはやなくなった後であって且つコンテナアプリケーションが閉じる前にコンテニーオブジェクトを完全にデアクチベートする。この方法は、コンテニーオブジェクトをその位置でアクチベートすることに関連したリソースの最終的な割り当て解除を実行する。図20は、IOLEInPlaceObject::InPlaceDeactivate方法を具現化するフローチャートである。この方法は、まず、そこに収容された（ネスト状）オブジェクトをアクチベートしたかどうかを決定し、もしそうであれば、ネスト状オブジェクトのInPlaceDeactivate方法を呼び出す。さもなければ、オブジェクトはそれ自身でデアクチベートする。ステップ2001において、この方法は、特定されたオブジェクトがコンテナオブジェクトでもありそしてネスト状のコンテニーオブジェクトをアクチベートしてい

るかどうか判断する。もしそうであれば、この方法は、ステップ2002へ続き、さもなくば、ステップ2004へ続く。ステップ2002において、この方法はアクチベートされたコンテニーオブジェクト（特定されたオブジェクトのサーバアプリケーションが既に記憶している）のIOLEInPlaceObjectインターフェイスを検索し、そしてステップ2003において、その検索されたインターフェイスのIOLEInPlaceObject::InPlaceDeactivate方法呼び出し、そして復帰する。ステップ2004において、この方法は、特定されたオブジェクトのユーザインターフェイスがまだアクティブであるかどうかを調べるようにチ

ェックし、もしそうであれば、ステップ2005へ続き、さもなくば、ステップ2006へ続く。ステップ2005において、この方法は、指定されたオブジェクトのInPlaceUIDeactivate方法呼び出してそれ自身のユーザインターフェイスをデアクチベートし、次いで、ステップ2006へ続く。ステップ2006において、この方法は、サーバアプリケーションファンクションRemove_Menuを呼び出して、複合メニューバーからサーバアプリケーションメニューを除去する。ステップ2007において、この方法は、特定されたオブジェクトのIOLEInPlaceFrame::RemoveMenus方法呼び出して、親コンテナアプリケーションがそのメニューを複合メニューバーから除去できるようにする。ステップ2008において、この方法は、オブジェクトリンク及び埋め込みAPIファンクションObjectDestroySharedMenuを呼び出し、複合メニューバーの構造を割り当て解除する。このObjectDestroySharedMenuファンクションは、ウインドウシステムに特定のもので、複合メニューバーに関連した構造を割り当て解除するためにその基礎となるウインドウシステムファンクションが必要とするものを呼び出す。ステップ2009において、この方法は、その基礎となるウインドウシステムファンクションDestroyMenuを呼び出し、複合メニューバー構造を割り当て解除すると共に、特定されたオブジェクトに関連したウインドウを割り当て解除する。最終的に、ステップ2010において、この方法は、特定されたオブジェク

トのコンテナオブジェクトのIOLEInPlaceParent::OnDeactivate方法呼び出し、そして復帰となる。

4. 5. 2 IOLEInPlaceObject::InPlaceUIDeactivate

InPlaceUIDeactivate方法は、その位置でアクチベートされた特定のオブジェクトに関連した全てのユーザインターフェイスエレメントを隠す。この方法は、複合文書内の異なるオブジェクト又はエリアのユーザ選択を処理するときにコンテナオブジェクトによって呼び出されるか、又は特定のオブジェクトのユーザインターフェイスがまだデアクチベートされていない場合に特定のオブジェクトのInPlaceDeactivateファンクション（図20参照）から呼び出される。図21は、IOLEInPlaceObject::InPlaceUIDeactivate

方法を具現化するフローチャートである。この方法は、まず、特定のオブジェクトがコンテナオブジェクトであってネスト状オブジェクトをアクチベートしているかどうかを決定し、もしそうならば、ネスト状オブジェクトのInPlaceUIDeactivateファンクションを呼び出す。さもなくば、この方法は、それ自身のユーザインターフェイスを隠し、そのユーザインターフェイスをデアクチベートしたことをそのコンテナアプリケーションに知らせる。ステップ2101において、この方法は、ネスト状コンテナオブジェクトがその位置でアクチベートされたことを指示するフラグが真であるかどうかを決定し、もしそうであれば、ステップ2102へ続き、さもなくば、ステップ2104へ続く。ステップ2102において、この方法は、アクチベートされたネスト状コンテナオブジェクトに対するIOLEInPlaceObjectインターフェイスを検索し、そしてステップ2103において、ネスト状コンテナオブジェクトのInPlaceDeactivate方法呼び出し、そして復帰となる。ステップ2104において、この方法は、フラグABOUT_TO_ACTIVATEをクリアし、ユーザが異なるオブジェクトを選択したことを示す。ステップ2105において、この方法は、特定のオブジェクトのアクチベート方法呼び出し、偽のパラメータを送って、デアクチベートする方法を要求する。こ

の方法は、親コンテナアプリケーションのフレームウィンドウに関連した全ての特定のオブジェクトのユーザインターフェイスエレメントを除去する。ステップ2106において、この方法は、オブジェクトリンク及び埋め込みAPIファンクションSetActiveObjectHwndを呼び出し、特定のオブジェクトのIOLEInPlaceObjectインターフェイスを、親コンテナアプリケーション文書ウィンドウとの関連から除去する。これは、コンテナアプリケーションがMDIアプリケーションである場合、及びユーザが後でこの文書ウィンドウを選択する場合に、特定のオブジェクトがその位置でもはや再アクチベートされないことを意味する。ステップ2107において、この方法は、その基礎となるウィンドウシステムファンクションを用いて、親コンテナアプリケーションの区画又は文書ウィンドウに関連したサーバアプリケーションに属するユーザインターフェイスエレメントを隠す。ステップ2108において、この

方法は、特定のオブジェクトのIOLEInPlaceParent::ShadeBorder方法を呼び出して、デアクチベートするオブジェクトの周りから陰影付けされたボーダーパターンフィードバックを除去する。ステップ2109において、この方法は、その基礎となるウィンドウシステムファンクションを呼び出し、特定のオブジェクトに関連したウィンドウを隠す。最後に、ステップ2110において、この方法は、IOLEInPlaceParent::OnUIDeactivate方法を呼び出し、コンテナアプリケーションがそれ自身のユーザインターフェイスをインストールし、そして復帰する。

4. 5. 3 IOLEInPlaceObject::TranslateAccelerator

TranslateAccelerator方法は、コンテナアプリケーションがアクセラータキーの組合せを処理する機会をもつ前にサーバアプリケーションがそれらの組合せを処理できるようにする。アクセラータキーの組合せは、メニューコマンドに対するキーボードショートカットであり、以下に詳細に説明する。本発明の好ましい実施例では、その位置でアクチベートされるオブジェクトは、規定により、先ずアクセラレータキーの組合せを処理する。TranslateAccelerator方法は、そのメッセージポンプにおいてコンテナアプリケーションによって呼び出される（コードテーブル9を参照された

い)。この方法によって実行される必要のある唯一のオペレーションは、特定のサーバアプリケーションのアクセラレータテーブルでその基礎となるウインドウシステムファンクション `TranslateAccelerator` を呼び出すことである。このような呼び出しは、コンテナーオブジェクトが個別の実行可能なプロセスによって具現化される場合には必要とされない。というのは、個別のプロセスは、これらキーの組合せをそれ自身のメッセージポンプにおいて受け取るが、コンテナーアプリケーションはそれらを決して受け取らないからである。この場合に、`TranslateAccelerator` 方法は何も行わない。

4. 5. 4 `IOLEInPlaceObject::Activate`

`Activate` 方法は、指定されたフラグ `fActive` が真であるか偽であるかに基づいて、親コンテナアプリケーションのフレームウインドウにインストールされたユーザインターフェイスエレメントをアクチベート又はデアクチベートする。MDI 文書ウインドウがアクチベート又はデアクチベートされたとき

に呼び出された場合には、この方法は、その位置でアクチベートされたオブジェクトに関連した複合メニューバーをインストールするか又は除去し、そして特定のオブジェクトがアクチベートされた場合にはその周りに陰影付けされたボーダーパターンを配する。最上レベルのフレームウインドウがアクチベート又はデアクチベートされたときに呼び出された場合には、この方法は、特定のオブジェクトがアクチベートされた場合にその周りに陰影付けされたボーダーパターンを配し、さもなければ、それを除去する。この場合に、他のユーザインターフェイスエレメントをアクチベート又はデアクチベートする必要はない。図 22 は、`IOLEInPlaceObject::Activate` 方法の具現化を示すフローチャートである。ステップ 2201 において、この方法は、最上レベルフレームウインドウのアクチベート又はデアクチベート動作の結果として呼び出されたか、或いは MID (子) 文書ウインドウのアクチベート又はデアクチベート動作の結果として呼び出されたかを決定する。MID 文書ウインドウのアクチベート又はデアクチベート動作の結果として呼び出された場合には、この方法は、ステップ 2202 へ進み、さもなければ、ステップ 2210 へ続く。ステップ 2202 において、この方法は、特定の

オブジェクトをアクチベートすべきかどうかを決定し、もしそうでなければ、ステップ2203へ続き、さもなければ、ステップ2206へ続く。ステップ2203において、この方法は、親コンテナオブジェクトのIOLEInPlaceFrame::SetMenu方法呼び出し、その位置での特定のオブジェクトのアクチベーションに関連した複合メニューバーを除去する。ステップ2204において、この方法は、親コンテナのアプリケーションフレームウィンドウにインストールされたユーザインターフェイスエレメントを隠す。ステップ2205では、この方法は、親コンテナオブジェクトのIOLEInPlaceParent::ShadeBorder方法呼び出し、特定のオブジェクトの周りから陰影付けされたボーダーパターンを除去し、復帰となる。ステップ2206において、この方法は、親コンテナオブジェクトのIOLEInPlaceFrame::SetMenu方法呼び出し、複合メニューバーを関連フレームウィンドウのメニューバーとしてインストールする。ステップ2207において、この方法は、コンテナアプリケーションのフレームウィンドウのタイトルバーをセットして、コンテナアプリケーションが特定のオブジェクトをアクチベ

ートしたことを指示する。ステップ2208において、この方法は、その基礎となるウィンドウシステムファンクションを呼び出し、フレームレベルのユーザインターフェイスエレメントを表示する。ステップ2209において、この方法は、親コンテナオブジェクトのIOLEInPlaceParent::ShadeBorder方法呼び出し、特定のオブジェクトの周りに陰影付けされたボーダーパターンを描いて、その位置でアクチベートされたことを指示し、次いで、復帰となる。ステップ2210において、この方法は、特定のオブジェクトをアクチベートすべきであるかどうかを決定し、もしそうであれば、ステップ2209へ続き、さもなければ、ステップ2211へ続く。ステップ2211において、この方法は、特定のオブジェクトの周りから陰影付けされたボーダーパターンを除去し、復帰となる。

4. 5. 5 IOLEInPlaceObject::ResizeBorder

ResizeBorder方法は、コンテナアプリケーションによって呼び出され、サーバアプリケーションが親コンテナアプリケーションの区画又は文書ウィンドウ内に配置したユーザインターフェイスツールをサイズ変更するように

サーバアプリケーションに要求する。この方法が呼び出されるのに応答して、サーバアプリケーションは、区画又は文書ウィンドウに関連したインターフェイスインスタンスの `QueryBorderSpace` 及び `SetBorderSpace` 方法を用いてコンテナアプリケーションとの別のツール配置交渉ループを開始しなければならない。

4. 5. 6 `IOLEInPlaceObject::EnableModeless`

`EnableModeless` 方法は、サーバアプリケーションに対して現在表示されているモードレスダイアログをイネーブル又はディスエイブルする。典型的に、この方法は、図16を参照して上記した `IOLEInPlaceFrame::EnableModeless` と同様に具現化される。

4. 5. 7 `IOLEInPlaceObject::SetVistRect`

`SetVistRect` 方法は、最も内側のレベルのコンテナオブジェクトによって呼び出され、実際に目に見えるオブジェクトの量を通信する。オブジェクトの目に見える（切り取り）四角形は、例えば、ボーダーの交渉、スクロール又はサイジングによって変更されてもよい。指定された四角形は、切り取り四角

形であり、コンテニーオブジェクトのウィンドウを正しい（切り取った）目に見えるサイズにサイズ変更するのはサーバアプリケーションの役目である。

4. 6 その他のサーバアプリケーションファンクション

好ましい実施例において、サーバアプリケーションは、次の1組のファンクションを与える。即ち、`ActivateUI:CreateNewMenu:CreateObjectToolBars:` 及び `RemoveMenu`。

4. 6. 1 `ActivateUI`

```
SCODE ActivateUI (IOLEInPlaceUIWindow*pDoc, IOLEInPlaceObject
pObject)
```

`ActivateUI` ファンクションは、指定されたコンテニーオブジェクトのユーザインターフェイスリソースのアクチベーションを制御するためにサーバアプリケーションによって具現化されるファンクションである。この高レベルファンクションは、フレーム、文書及び区画レベルのユーザインターフェイスエレ

メントをアクチベートし、オブジェクトの周りに陰影付けされたボーダーパターンを描き、そして複合メニューバーを表示する。図23は、ActivateUIファンクションを具現化するフローチャートである。このファンクションは、2つのパラメータ、即ち文書インターフェイスに対するポインタと、コンテニーオブジェクトに対するポインタとを取り出す。ステップ2301において、このファンクションは、指定された文書ウインドウ及び指定されたコンテニーオブジェクトに対するウインドウハンドルを得る。ステップ2302において、このファンクションは、オブジェクトリンク及び埋め込みAPIファンクションSetActiveObjectHwndを呼び出し、指定された文書ウインドウの現在アクティブオブジェクトを、コンテニーオブジェクトのインターフェイスに対するポインタにセットする。これは、文書ウインドウの1つがユーザにより選択されたときに、MDIアプリケーションとして具現化されるコンテナアプリケーションが適切なコンテニーオブジェクトをアクチベートできるようにする。ステップ2303では、このファンクションは、IOLEInPlaceObject::Activate方法を呼び出し、指定されたオブジェクトをアクチベートする。ステップ2304では、このファンクションは、その基礎となるウインドウシステムファンクショ

ンShowWindowを呼び出し、コンテナアプリケーションの区画又は文書ウインドウに関連したユーザインターフェイスエレメントを表示する。ステップ2305において、このファンクションは、指定されたコンテニーオブジェクトを取り巻くためのボーダー又は四角形の寸法を決定し、そしてステップ2306において、このファンクションは、指定されたコンテニーオブジェクトのIOLEInPlaceObject::ShadeBorder方法を呼び出し、この四角形を用いて指定されたコンテニーオブジェクトの周りに陰影付けされたボーダーパターンを描く。ステップ2307において、このファンクションは、入力フォーカスを、指定されたコンテニーオブジェクトのオブジェクトウインドウにセットする。最終的に、ステップ2308において、このファンクションは、その基礎となるウインドウシステムファンクションDrawMenuBarを呼び出して、複合メニューバーを再表示し、そして復帰となる。

4. 6. 2 CreateNewMenu

HANDLE CreateNewMenu (IOLEInPlaceFrame *pFrame)

Code Table 2

```

HANDLE CreateNewMenu (IOLEInPlaceFrame *pFrame) {
1  hmenu = CreateMenu ( );
2  pFrame -> InsertMenus (hmenu, ContrCounts);
3  lpiMenuCount[0] = ContrCount[0];
4  lpiMenuCount[2] = ContrCount[1];
5  lpiMenuCount[4] = ContrCount[2];
6  if there are Edit group menus present {
7      for each editgroupmenu {
8          heditmenu = CreateMenu ( );
9          insertion_point = lpiMenuCount[0] + lpiMenuCount[1] + 1;
10         InsertMenu (hmenu, insertion_point, MF_BYPOSITION | MF_POPUP,
11                     heditmenu, NULL);
12         lpiMenuCount[1] = lpiMenuCount[1] + 1;
13         for each editgroupmenu_item {
14             InsertMenu (heditmenu, -1, MF_BYPOSITION | MF_STRING, item_id,
15                         "string to be displayed");
16         }
17     };
18  if there are Object group menus present {
19      for each objectgroupmenu {
20          hobjmenu = CreateMenu ( );
21          insertion_point = lpiMenuCount[0] + lpiMenuCount[1] + lpiMenuCount[2]
22          +
23          lpiMenuCount[3] + 1;
24          InsertMenu (hmenu, insertion_point, MF_BYPOSITION | MF_POPUP,
25                      hobjmenu, NULL);
26          lpiMenuCount[3] = lpiMenuCount[3] + 1;
27          for each objectgroupmenu_item {
28              InsertMenu (hobjmenu, -1, MF_BYPOSITION | MF_STRING, item_id,
29                          "string to be displayed");
30          }
31  };
32  if there are Help group menus present {
33      for each helpgroupmenu {
34          hhhelpmenu = CreateMenu ( );
35          InsertMenu (hmenu, -1, MF_BYPOSITION | MF_POPUP, hhhelpmenu,
36                      NULL);
37          lpiMenuCount[5] = lpiMenuCount[5] + 1;
38          for each objectgroupmenu_item {
39              InsertMenu (hhhelpmenu, -1, MF_BYPOSITION | MF_STRING,
40                          item_id,
41                          "string to be displayed");
42          }
43  };
44  hSharedMenu = ObjectCreateSharedMenu (hmenu, lpiMenuCount);
45  return (hSharedMenu);
46  }

```

CreateNewMenuファンクションは、複合メニューバーの形成を管理するためにサーバアプリケーションによって具現化されるファンクションである

る。このファンクションは、複合メニューバーに関連した構造を割り当て、そのメニューを挿入するようにコンテナアプリケーションを要求し、そしてサーバアプリケーションメニューを挿入する。コードテーブル2は、CreateNewMenuファンクションの具現化を表している。ライン1において、このファンクションは、その基礎となるウインドウシステムファンクションを呼び出し、複合メニューバーに対するデータ構造を形成する。ライン2において、このファンクションは、コンテナアプリケーションのフレームウインドウのIOLEInPlaceFrame::InsertMenusを呼び出し、コンテナアプリケーションのメニューを複合メニューバーに挿入する。ライン3ないし5において、このファンクションは、各メニューグループに対してコンテナアプリケーションが挿入したメニューの数を追跡する。ライン6ないし17において、サーバアプリケーションがEditグループメニューを有すると仮定すれば、このファンクションは、このファンクションは、各Editグループメニューを形成し、そしてそれを複合メニューバーにおける正しいスポットに挿入し、いかに多くのメニューが挿入されたかを追跡する。正しいスポットは、ライン9において、いかに多くのメニューが既に左側へ挿入されたかを決定することにより計算される。これは、Editグループに対し、コンテナアプリケーションがコンテナグループの一部として挿入したメニューの数と、既に挿入されているEditグループのメニューの数と、現在挿入に対する1との和となる。ライン18ないし30、31ないし41において、このファンクションは、各々Objectグループ及びHelpグループに属するメニューに対して同様のステップを実行する。ライン42において、このファンクションは、オブジェクトリンク及び埋め込みAPIファンクションObjectCreateSharedMenuを呼び出して、複合メニューバーに対するメッセージハンドリングに関連したデータ構造を形成し、そしてライン43において、ハンドルをこの構造に返送する。

4. 6. 3 CreateObjectToolbars

```
void CreateObjectToolbars(IOLEInPlaceFrame*pFrame,
    IOLEInPlaceUIWindow*pDoc, IOLEInPlaceUIWindow*pPane)
```

CreateObjectToolbarsファンクションは、サーバアプリケーションツールに必要なスペースに対しサーバアプリケーションとコンテナアプリケーションとの間で交渉するためにサーバアプリケーションによって具現化されるファンクションである。図24は、CreateObjectToolbarsファンクションを具現化するフローチャートである。ステップ2401ないし2408は、サーバアプリケーションに対するツールを形成するように必要に応じて何回も繰り返される。ステップ2401において、このファンクションは、形成すべきツールバーが更にあるかどうかを決定し、もしなければ、復帰し、さもなければ、ステップ2402に続く。ステップ2402において、このファンクションは、（サーバアプリケーションがツールをどこに配置したいかに基づいて）コンテナアプリケーションのフレーム、文書又は区画ウインドウのIOLEInPlaceUIWindow::GetBorder方法呼び出し、交渉を開始する。ステップ2403において、このファンクションは、所望のフレーム、文書又は区画ウインドウのIOLEInPlaceUIWindow::QueryBorderSpace方法呼び出し、GetBorder方法に対する以前の呼び出しにより返送された四角形の内側にボーダースペースの特定の巾を要求する。これらの方法は、以下に詳細に説明する。ステップ2404において、特定の巾を受け入れられない場合には、このファンクションはステップ2405に続き、さもなければ、復帰となる。異なる値で必要な回数だけQueryBorderSpaceを呼び出すことにより異なる量のスペースに対して交渉するように具現化を選択することができる。ステップ2405において、このファンクションは、ステップ2403で既に交渉されているスペースでIOLEInPlaceUIWindow::SetBorderSpace方法呼び出す。ステップ2406において、このファンクションは、親コンテナアプリケーションの所望のフレーム、文書又は区画ウインドウの子ウインドウを形成する。ステップ2407において、このファンクションは、既に形成した子ウインドウにツールを描き、次いで、ループの始めに復帰する。

4. 6. 4 RemoveMenus

```
void RemoveMenus(HANDLE hSharedMenu)
```

Code Table 3

```

RemoveMenus(HANDLE hSharedMenu) {
1  menu = hSharedMenu -> menu;
2  descriptor = hSharedMenu -> descriptor;
3  for (i = descriptor[0] + 1, i <= descriptor[1]. i++) {
4      RemoveMenu (menu, heditmenu, MF_BYPOSITION)
5  }
6  for (i = descriptor[2] + 1, i <= descriptor[3]. i++ {
7      RemoveMenu (menu, hobjmenu, MF_BYPOSITION)
8  }
9  for (i = descriptor[4] + 1, i <= descriptor[5]. i++ {
10     RemoveMenu (menu, hhelptmenu, MF_BYPOSITION)
11 }
12 return ( );
}

```

RemoveMenusファンクションは、コンテニーオブジェクトのデアク
チベーション時に複合メニューバーからサーバアプリケーションのメニューを除
去する。この方法は、コンテニーオブジェクトのInPlaceDeactiv
ate方法から呼び出される。コードテーブル3は、RemoveMenusフ
ァンクションの具現化を示す。このファンクションは、各メニューグループ内に
多数のメニューを含む共用メニュー記述子（図25参照）に記憶された情報を用
いて、その基礎となるウインドウシステムファンクションRemoveMenu
を呼び出すことにより全てのサーバアプリケーションメニューを除去する。ライ
ン3ないし5は、Editグループに属するメニューを除去し、ライン6ないし
8は、Objectグループに属するメニューを除去し、そしてライン9ないし
11は、Helpグループに属するメニューを除去する。

4. 7 オブジェクトリンク及び埋め込みAPIのヘルパーファンクション

インターフェイス定義に加えて、オブジェクトリンク及び埋め込みAPIは、
コンテナー及びサーバアプリケーションによって使用されるべき1組のヘルパー
ファンクションを与える。これらのファンクションは、SetActiveOb
jectHwnd; GetActiveObjectHwnd; ObjectC
rcatcSharedMenu; ObjectDestroySharedM

enu: ObjectShade; 及びObjectSetMenuを含む。

4. 7. 1 SetActiveObjectHwnd

```
void SetActiveObjectHwnd(HWND hwndDOC,  
IOLEInPlaceObject*pObject)
```

SetActiveObjectHwndファンクションは、MDIアプリケーションの現在選択されたオブジェクトをセットする。MDIコンテナアプリケーションの各MDI（文書）ウィンドウには、そのMDIウィンドウが最後に入力フォーカスを有したときに表示されたその位置でアクチベートされたオブジェクトに対応するオブジェクトインターフェイスが関連される。MDIウィンドウ内からオブジェクトがその位置でアクチベートされていない場合には、それに関連したオブジェクトインターフェイスはNULLである。このメカニズムは、MDIウィンドウが後で入力フォーカスを受け取る時に、例えば、ユーザがMDIウィンドウのタイトルバーにおいてマウスでクリックするときに、適切なコンテナオブジェクトをアクチベートするようにMDIウィンドウをイネーブルする。（ユーザが他のMDIウィンドウのタイトルバーにおいてクリックするときは、第1のウィンドウに関連したいかなるイン・プレース対話も表示から消失する。）SetActiveObjectHwndファンクションは、2つのパラメータ、即ちMDI（文書）ウィンドウのウィンドウハンドルと、その位置で現在アクチベートされているオブジェクトのIOLEInPlaceObjectインターフェイスとを取り上げる。ウィンドウハンドルをオブジェクトインターフェイスに関連させる方法は多数あることが当業者に明らかであろう。1つの実施例において、このファンクションは、オブジェクトインターフェイスを、その基礎となるウィンドウシステムファンクションを用いて文書ウィンドウの特性として記憶する。この具現化は、システムにおいてアクティブな各MDIウィンドウごとに記憶装置を必要とすることに注意されたい。別の解決策は、現在選択されるオブジェクトを追跡する方法を文書、区画及びフレームウィンドウインターフェイスに追加する。

4. 7. 2 GetActiveObjectHwnd

```
HWND GetActiveObjectHwnd(HWND hwndDOC)
```

GetActiveObjectHwndファンクションは、MDIウィンド

ウが入力フォーカスを受け取るときにその位置でアクチベートされるべきコンテニーオブジェクトを検索する。このファンクションは、SetActiveObjectHwndファンクションを用いて既に記憶されているオブジェクトインターフェイスを返送する。

4 7. 3 ObjectCreateSharedMenu

```
HANDLE ObjectCreateSharedMenu(HMENU hMenuCombined,
UINT lpiMenuCounts)
```

Code Table 4

```
HANDLE ObjectCreateSharedMenu (hmenu, lpiMenuCount) {
1: hSharedMenu = AllocateSharedMenuHandle();
2: hSharedMenu -> menu = hmenu;
3: for (i = 0; i < number_menu_groups; i++) {
4:     if (i > 0) {
5:         hSharedMenu -> descriptor[i].count =
6:             hSharedMenu -> descriptor[i - 1].count + lpiMenuCount[i];
7:         if i is even
8:             hSharedMenu -> descriptor[i].function = Id_Container
9:         else hSharedMenu -> descriptor[i].function = Id_Object;
10:     }
11:     else {
12:         hSharedMenu -> descriptor[0].count = lpiMenuCount[0];
13:         hSharedMenu -> descriptor[0].function = Id_Container;
14:     };
15: }
16: return (hSharedMenu);
}
```

ObjectCreateSharedMenuファンクションは、その位置でアクチベートされたオブジェクトの複合メニューバーに関連した共用メニューデータ構造を形成する。このファンクションは、コンテニーオブジェクトがアクチベートされたときにCreateNewMenuファンクションから呼び出される。コードテーブル4は、ObjectCreateSharedMenuファンクションの具現化を示す。このファンクションは、2つのパラメータ、即ち複合メニューバーに対するハンドルと、各メニューグループにおけるメニューの

数を含むメニューカウントのアレーとを取り上げる。このファンクションは、新たに形成された共用メニューデータ造に対するハンドルを返送する。ライン1にお

いて、このファンクションは、共用メニューデータ構造に必要なメモリを割り当てる。ライン2において、このファンクションは、このデータ構造における複合メニューバーに対するハンドルをセーブする。ライン3ないし15では、このファンクションは、メニューカウントアレーに記憶されたファンクションに基づいて共用メニュー記述子を設定する。この記述子は、メニューコマンドを受け取るときにウインドウ手順によって使用され、メニューコマンドをコンテナアプリケーションに送るべきかサーバアプリケーションに送るべきかを決定する。

この情報を維持する種々の方法があることが当業者に明らかであろうが、1つの実施例では、記述子は、各インデックスにおいて、そのインデックスに関連したメニューグループに含まれた最後のメニューの数を記憶する。(メニューは、左側において1から番号付けされる。)又、メニューグループがコンテナアプリケーションに属するかサーバアプリケーションに属するかの指示も各インデックスに記憶される。又、どのメニューグループに対してどのアプリケーションに通知すべきかを指示する付加的なパラメータを通すことにより、メニューグループ分け機構もサポートできることが当業者に明らかであろう。この記述子構成を用いて、ウインドウ手順は、メニュー項目選択を含むまでメニューの数をカウントしそしてそのメニュー数を記述子の値と比較して正しいインデックスを見つけることにより、特定のメニュー項目選択がどのインデックス内に入るかを決定することができる。いったんインデックスが決定されると、ウインドウ手順は、コンテナアプリケーションファンクションを呼び出すべきか又はサーバアプリケーションファンクションを呼び出すべきかを指定する指示子を検索することができる。この手順は、コードテーブル5を参照して以下に詳細に述べる。

図25は、図4について述べた例に対応する共用メニューデータ構造のブロック図である。この共用メニューデータ構造は、複合メニューバーのポインタ2502と、各メニューグループに対するメニューカウントを含む記述子2505とで構成される。複合メニューバー2503は、コンテナ及びサーバアプリケーション2504からのメニューを含む。記述子2505の各エレメントは、カウントフィールド2506及びファンクションフィールド2507を有する。カウ

ントフィールド2506は、メニューグループ内の最後のメニューの数（左からスタートして）を指示する。例えば、第2のメニューグループは、Editグループであり、1つのメニューしか含まない。このメニュー2503は、複合メニューバーにおいて左から2番目のメニューであり、それ故、カウントフィールド2509は、数2を含む。別の例として、4番目のメニューグループは、Objectグループである。このグループは、サーバアプリケーション2510からの5つのメニューを含む。それ故、このメニューグループ2511のカウントは数7を含む。というのは、7番目のメニューが、Objectグループにおける最後のメニューであるMacroメニューだからである。

4. 7. 4 ObjectDestroySharedMenu

```
void ObjectDestroySharedMenu(HMENU hMenuCombined)
```

ObjectDestroySharedMenu関数は、ObjectCreateSharedMenuへの以前の呼び出しにおいて形成された共用メニューデータ構造を破壊する。この関数は、コンテナー及びサーバアプリケーションがそれらのメニューを複合メニューバーから除去した後、アクチベートされたコンテナーオブジェクトのIOLEInPlaceObject::InPlaceDeactivate方法から呼び出される。

4. 7. 5 ObjectShade

```
void ObjectShade(HWND hwndParent, LPRECT lprc, DWORD grfState)
```

ObjectShade関数は、オブジェクトリンク及び埋め込みAPIによって与えられ、その位置でアクチベートされたオブジェクトの周りに配置される陰影付けされたボーダーパターンを形成する。hwndParentパラメータは、アクチベートされた（又はアクチベートされるべき）オブジェクトのIOLEInPlaceParentインターフェイスに関連したウィンドウハンドルである。lprcパラメータは、パターンが配置されるところの親ウィンドウ座標における四角形である。grfStateフラグは、IOLEInPlaceParent::ShadeBorder方法について述べたものと同じであり、SHADEBORDER_ON及びSHADEBORDER_ACTIVEを含む。

4. 7. 6 ObjectSetMenuDescriptor

```
SCODE ObjectSetMenu(HWND hwndFrame,HOLEMENU hMenuCombine,  
    HWND hwndObject)
```

ObjectSetMenuDescriptorファンクションは、複合メニューバーに対するメッセージハンドラーを設定又は除去する。このファンクションは、アクチベートするコンテナオブジェクトの関連コンテナオブジェクトのIOLEInPlaceFrame::SetMenu方法によって呼び出される。図26は、ObjectSetMenuDescriptorファンクションの具現化を示すフローチャートである。これは、3つのパラメータ、即ちコンテナアプリケーションに関連したフレームウインドウのウインドウハンドル: ObjectCreateSharedMenuファンクションによって返送された共用メニューデータ構造に対するハンドル: 及び現在その位置でアクチベートされるべきオブジェクトのウインドウハンドルを取り上げる。共用メニュー構造に対するハンドルがナルである場合には、このファンクションは、複合メニューバーに対するメッセージハンドラーを除去し、さもなければ、メッセージハンドラーを設定する。ステップ2601において、このファンクションは、指定された共用メニューデータ構造に対するハンドルがナルであるかどうか決定し、もしそうであれば、ステップ2602へ続き、さもなければ、ステップ2603へ続く。ステップ2602において、このファンクションは、その基礎となるウインドウシステムファンクションSetWindowLongを呼び出し、コンテナアプリケーションに既に関連されている特殊なメッセージハンドラーを除去する。ステップ2603において、このファンクションは、複合メニューに対して既に設定されている特性を除去し、次いで、復帰する。ステップ2604において、このファンクションは、特殊なメッセージハンドラーによつて後で使用されるべき共用メニューデータ構造を記憶するための特性をフレームウインドウにセットする。ステップ2605において、このファンクションは、アクチベートするオブジェクトのウインドウハンドルに対応する別の特性をフレームウインドウにセットする。ステップ2606において、このファンクションは、その基礎となるウインドウシステムファンクションSetWindowLongを用いて、親コンテナアプリケーション

シ

ョンのフレームウインドウに対する新たなウインドウ手順として特殊なメッセージハンドラーをインストールする。古いウインドウ手順は、特性Old_Filterに後で使用するためにセーブされる。(例えば、以下に詳細に述べるコードテーブル5を参照されたい。)次いで、ファンクションは復帰となる。

5. イン・プレース対話APIの使用

イン・プレース対話をサポートするオブジェクトリンク及び埋め込みAPIファンクションは、次のことを行うために呼び出される。

- ・SDI又はMDIコンテナアプリケーション内の場所でオブジェクトをアクチベートする。

- ・ユーザがコンテナアプリケーションの複合メニューバーからメニュー項目を選択するときにメッセージを処理する。

- ・ユーザが異なるオブジェクトをアクチベートするように選択するか又は最上レベルのコンテナアプリケーションをその通常処理に回復するよう選択するときにその位置でアクチベートされたオブジェクトのユーザインターフェイスリソースをデアクチベートする。

- ・コンテナアプリケーションがイン・プレース対話リソースをもはや必要としないときにサーバアプリケーションに対してそれらリソースをデアクチベートする。

- ・サーバアプリケーションがダイアログを表示しそしてコンテナアプリケーションがモードレスダイアログを現在表示する(又はその逆である)ときにモードレスダイアログをイネーブル及びディスエイブルする。

- ・アクセラレータキー組合せをコンテナアプリケーションとサーバアプリケーションとの間に分配するようにこれら組合せを処理する。

5. 1 その位置でアクチベーションする手順

上記したように、オブジェクトが文書にリンクされるか又は埋め込まれると、ユーザは、オブジェクトを選択し、そしてその選択されたオブジェクトに基づいてあるアクションを実行するよう要求する。図3及び4の例に戻ると、ユーザが

スプレッドシートオブジェクト305をその位置でアクチベートするように望む場合には、ユーザは、オブジェクトプレゼンテーションフォーマットにおいてマ

ウス入力装置で2回クリックするか、又はコンテナアプリケーションメニューを用いてオブジェクトにおけるアクションを選択することができる。図8は、ユーザがメニューを使用してスプレッドシートオブジェクト305をその位置でアクチベートできるようにする1つの方法を示している。ユーザがコンテナアプリケーション（ワードプロセッサアプリケーション）のEditメニュー802からメニュー項目「Excel Worksheet Object」803を選択し、そして「Excel Worksheet Object」サブメニュー804からいずれかのアクションを選択するときには、ワードプロセッサアプリケーションは、スプレッドシートアプリケーションを呼び出して、スプレッドシートオブジェクトをその位置でアクチベートする。

スプレッドシートオブジェクト305をアクチベートするプロセスは、多数のステップで行われる。第1に、ワードプロセッサアプリケーションのフレームウィンドウに対するウィンドウ手順が、その基礎となるウィンドウシステムにより、オブジェクトアクションサブメニュー804におけるメニュー項目のユーザ選択に応答して呼び出される。（例えば、図14を参照されたい。）第2に、メニュー事象を受け取ると、ウィンドウ手順は、ファンクションProcess__Object__Activationを呼び出す。（例えば、ステップ1407、1411及び1413を参照されたい。）第3に、ファンクションProcess__Object__Activationは、オブジェクトリンク及び埋め込みAPIのファンクションObjectLoadを用いてスプレッドシートオブジェクト305に対するデータをロードする。最後に、ファンクションProcess__Object__Activationプログラムは、スプレッドシートオブジェクト305のDoVerb方法と呼ばひ出して、選択されたアクションを実行するようにスプレッドシートアプリケーションに要求する。

図27は、ファンクションProcessObjectActivationを具現化するフローチャートある。このファンクションは、選択されたオブジェ

クトをロードし、そしてそのDoVerb方法を呼び出して、選択されたアクションを実行する。ステップ2701において、このファンクションは、オブジェクトリンク及び埋め込みAPIのファンクションObjectLoadを

呼び出し、オブジェクトの記憶に対するポインタと、それがIOLEObjectインターフェイスを希望するという指示とをそれに通す。ファンクションObjectLoadは、ロードされたオブジェクトのIOLEObjectインターフェイスに対するポインタを返送する。ステップ2702において、このファンクションは、コンテナーオブジェクトのSetClientSite方法を呼び出して、その関連する親コンテナーオブジェクトインターフェイス（pclientsite）に対するポインタをコンテナーオブジェクトに手渡す。ステップ2703において、このファンクションは、ロードされたオブジェクトのIOLEObject::DoVerb方法を呼び出し、選択されたアクション、そのアクションがダブルクリックによって選択されたかどうかの指示、及びオブジェクトに対して既に形成されているIOLEClientSiteインターフェイスをそこに通す。次いで、このファンクションは復帰となる。

図28は、オブジェクトリンク及び埋め込みAPIファンクションObjectLoadの具現化を示すフローチャートである。このファンクションは、オブジェクトのメモリ内インスタンスを形成し、将来の対話についてサーバアプリケーションを準備し、そして指定されたインターフェイスに対するポインタを返送する。このファンクションは、3つのパラメータ、オブジェクトデータがロードされるべき記憶装置に対するポインタ、発呼者が返送を望むインターフェイスの指示、及びオブジェクトのメモリ内インスタンスに対する返送ポインタを取り上げる。ステップ2801において、このファンクションは、CLASS_IDを指定の記憶装置から検索する。ステップ2802において、このファンクションは、検索されたCLASS_IDを用いて、この形式のオブジェクトのメモリ内インスタンスを形成するためのコードを位置決めする。本発明の好ましい実施例においては、IOLECreateインターフェイスが各サーバアプリケーションによって与えられ、それが具現化するオブジェクトのメモリ内インスタンスを

形成する。ステップ2803において、このファンクションは、IOLECreate::CreateInstance方法呼び出し、オブジェクトに対するメモリ内構造を形成し、そしてオブジェクトに対する永続的記憶装置をアクセスするところのIPersistStorageインターフェイスに対するポイン

タを返送する。ステップ2804において、このファンクションは、IPersistStorage::Load方法呼び出し、これは、指定された記憶装置からオブジェクトデータをロードする。ステップ2805において、このファンクションは、IPersistStorage::QueryInterface方法呼び出し、指定のインターフェイスを検索すると共に、その検索されたインターフェイスを返送する。

図29は、IOLEObject::DoVerb方法の典型的な具現化を示すフローチャートである。この方法は、コンテニーオブジェクトと対話するための主たる方法である。この方法は、コンテナーアプリケーションと交渉して、イン・プレース対話を行うと共に、サーバアプリケーションのユーザインターフェイスをアクチベートさせる。この方法は、4つのパラメータ、即ちユーザが選択したアクション；ユーザがアクションを選択したときにコンテナーアプリケーションのウインドウ手順によって受け取られるメッセージ構造に対するポインタ；オブジェクトのIOLEClientSiteインターフェイスに対するポインタ；及び例えば、サーバアプリケーションが呼び出し時に入力フォーカスを得るべきかどうかについてverb.の実行を制御する1組のフラグを取り上げる。ステップ2901において、この方法は、IOLEClientSite::QueryInterface方法呼び出して、特定のオブジェクトに対するIOLEInPlaceParentインターフェイスを得る。ステップ2902において、この方法は、IOLEInPlaceParent::CanInPlaceActivate方法呼び出して、コンテナーアプリケーションがイン・プレース対話をサポートするかどうかを決定する。ステップ2903において、コンテナーアプリケーションがイン・プレース対話をサポートしない場合には、この方法はステップ2904に続き、さもなければ、この方法はステップ29

06に続く。ステップ2904において、この方法は、サーバアプリケーションのフレームウインドウを形成し表示する。というのは、コンテナアプリケーションがイン・プレース対話を実行できないからである。ステップ2905において、この方法は、指定されたアクションの通常の処理を続け、復帰する。ステップ2906において、この方法は、`IOLEInPlaceParent::G`

`etWindowContext`方法呼び出し、コンテナアプリケーションに関連したインターフェイスを得る。ステップ2907において、この方法は、特定のオブジェクトでイン・プレース対話をサポートするために形成する必要のあるオブジェクトウインドウのサイズを計算する。ステップ2908において、この方法は、`IOLEInPlaceParent::GetWindowContext`方法により返送されたエリアがスケーリング又はクリッピングを必要とするかどうか及び特定のオブジェクトがこれをサポートできるかどうかを決定する。必要とされるサイズをサポートできる場合には、この方法はステップ2909に続き、さもなくば、この方法は、イン・プレース対話を放棄してステップ2909へ続く。ステップ2909において、この方法は、`IOLEInPlaceParent`インターフェイスに対応するウインドウに対するウインドウハンドルを検索する。ステップ2910において、この方法は、オブジェクトルートウインドウとして使用されるべき`IOLEInPlaceParent`インターフェイスに対応するウインドウの子として新たなウインドウを形成する。(例えば、図13の項目1309を参照されたい。) ステップ2911において、この方法は、特定のオブジェクトに対するユーザインターフェイスリソースがまだ使用できるかどうか、即ち割り当てられているがまだ割り当て解除されていないかどうかを決定する。これらリソースが使用できる場合には、この方法はステップ2913へ続き、さもなくば、この方法はステップ2912へ続く。ステップ2912において、この方法は、`IOLEInPlaceParent::OnUIActivate`方法呼び出し、コンテナアプリケーションが、特定のオブジェクトをその位置でアクチベートする準備においてそのユーザインターフェイスリソースを除去できるようにする。ステップ2913において、この方法

は、`IOLInPlaceParent::OnInPlaceActivate`方法呼び出し、コンテナアプリケーションが、ネスト状オブジェクトをその位置でアクチベートしたことを記録できるようにする。ステップ2914において、この方法は、ファンクション`CreateNewMenu`を呼び出し、新たな複合メニューバーを形成する（コードテーブル2を参照）。ステップ2915において、この方法は、特定のオブジェクトが付加的なユーザインターフェ

イスツールのアクチベーションを必要とするかどうかを決定し、もしそうであれば、ステップ2916へ続き、さもなければ、ステップ2917へ続く。ステップ2916において、この方法は、ファンクション`CreateObjectTooltbars`を呼び出し、特定のオブジェクトの付加的なユーザインターフェイスツールの位置を交渉しそしてそれを配置する（図24を参照）。ステップ2917において、この方法は、ファンクション`ActivateUI`を呼び出し、これは、特定のオブジェクトの全てのユーザインターフェイスリソースを表示させ（図23を参照）そして復帰となる。

5. 1. 1 多文書インターフェイスアプリケーション内の位置でのアクチベーション

オブジェクトがSDIコンテナアプリケーションからアクチベートされたと仮定してコンテニーオブジェクトのアクチベーションについて上記した。一方、オブジェクトがMDIコンテナアプリケーション（このアプリケーションは、定義により、多数の複合文書と同時に対話できる）内でアクチベートされる場合には、オブジェクトを含む文書（MDI）ウインドウがアクチベート又はデアクチベートされるときにアクチベーション及びデアクチベーションが生じる。文書ウインドウに対するウインドウ手順は、ユーザがウインドウを選択する（例えば文書ウインドウのタイトルバーにおいてクリックすることにより）ときに、その基礎となるウインドウシステムからアクチベーションメッセージを受け取る。文書ウインドウに対するウインドウ手順は、ユーザが次いで異なるウインドウを選択するときにデアクチベーションメッセージを受け取る。これらのメッセージに応答して、文書ウインドウに対するウインドウ手順は、ファンクション（例えば

Process_Activation_Message) を呼び出し、文書ウインドウ及びその中に含まれたアクチベートされたオブジェクトのアクチベーション及びデアクチベーションを実行する。

図30は、アクチベーション及びデアクチベーションメッセージを処理するためにMDI文書ウインドウのウインドウ手順によって呼び出されたファンクションProcess_Activation_Messageの具現化を示すフローチャートである。1つの実施例において、文書ウインドウハンドルがファンク

ションに対するパラメータとして通される。このファンクションは、ウインドウが最後にアクティブとなったときに既にその位置でアクチベートされたオブジェクトをウインドウが含むかどうかを決定する。もしそうであれば、ファンクションは、そのオブジェクトをアクチベート又はデアクチベートし、さもなければ、文書ウインドウをその通常の形態でアクチベート又はデアクチベートする。ステップ3002において、このファンクションは、その通常のウインドウデアクチベーション手順を実行し、そして復帰となる。ステップ3001において、このファンクションは、ファンクションGetActiveObjectHwndを呼び出すことにより、既にアクティブな収容オブジェクトがもしあればそれに対しIOLEInPlaceObjectのオブジェクトインターフェイスを検索する。ステップ3002において、このファンクションは、オブジェクトインターフェイスがナルであるかどうか決定する。もしナルであれば、既にアクティブな収容オブジェクトは存在せず、ファンクションはステップ3003へ続き、さもなければ、ファンクションはステップ3004へ続く。ステップ3003では、このファンクションは、文書ウインドウツール及びメニューを設定するといったその通常のウインドウアクチベーション又はデアクチベーション手順を実行し、そして復帰となる。ステップ3004では、このファンクションは、指定されたフラグFActiveが真であるかどうか決定する。フラグFActiveが真である場合は、既にアクティブな収容オブジェクトをアクチベートすべきであり、このファンクションはステップ3005に続き、さもなければ、オブジェクトをデアクチベートすべきであり、ファンクションはステップ3006に続く。ステッ

ブ3005では、このファンクションは、検索されたインターフェイスのIOL
EInlaceObject::Activate方法と呼ばひ出して、既にアク
チベートされたオブジェクトがそれ自体アクチベートするように要求し、そして
復帰となる。ステップ3006では、このファンクションは、検索されたインター
フェイスのIOLEInPlaceObject::Activate方法
と呼ばひ出して、既にアクチベートされたオブジェクトがそれ自体デアクチベートす
るように要求し、そして復帰となる。

5. 2 プルダウンメニューメッセージハンドリングのユーザ選択

ユーザがオブジェクトをその位置でアクチベートした後に、ユーザは、コンテ
ナーアプリケーションのメニューバー（これは複合メニューバーである）を介し
てアクションを選択することによりコンテナアプリケーション内のオブジェク
トと対話する。メニューのあるものはサーバアプリケーションに属しておりそし
て他のメニューはコンテナアプリケーションに属しているので、コンテナア
プリケーションのフレームウインドウのウインドウ手順は、メニュー入力事象を
コンテナアプリケーション内のファンクションに送信すべきかサーバアプリケ
ーション内のファンクションに送信すべきか判断しなければならない。このため
に、オブジェクトリンク及び埋め込みAPIのファンクションObjectSe
tMenuDescriptorがサーバアプリケーションによって呼び出され
たときにこのファンクションによって特殊なメッセージハンドラーがインストー
ルされ、新たに形成された複合メニューバーをインストールする。この特殊なメ
ッセージハンドラーは、これがインストールされると、コンテナアプリケーション
のフレームウインドウに対する新たなウインドウ手順となる。従って、その
フレームウインドウに対応するコンテナアプリケーションによって受け取られ
た全てのメッセージは、その後に、先ず、特殊なメッセージハンドラーへ送られ
る。この特殊なメッセージハンドラーは、次いで、受け取ったメッセージ事象を
どのアプリケーションへ送るべきかを判断する。

Code Table 5

```
InPlaceWndProc ( hwnd, message, wparam, lparam ) {  
1 /* wparam = item id of menu item selected ; */  
2 /* hiword (lparam) = hmenu containing the item */  
3 switch (message) {  
4     case WM_COMMAND:  
5         SetFocus (Old_Focus);  
6         if (saveMenuRoutine == Id_Object) {  
7             hwndObj = GetProp(hwndFrame, "InPlaceObject");  
8             PostMessage(hwndObj, message, wparam, lparam);  
9         }  
10        else {  
11            Old_Filter = GetProp(hwndFrame, "OldFilter");  
12            call Old_Filter (hwndFrame, message, wparam, lparam);  
}
```

```

13     };
14     break;
15     case WM_INITMENUPOPUP:
16         WM_ENTERIDLE.
17         WM_MEASUREITEM.
18         WM_DRAWITEM:
19         if (saveMenuRoutine == Id_Object) {
20             hwndObj = GetProp(hwndFrame, "InPlaceObject");
21             PostMessage(hwndObj, message, wParam, lParam);
22         }
23         else {
24             Old_Filter = GetProp(hwndFrame, "OldFilter");
25             call_Old_Filter (hwndFrame, message, wParam, lParam);
26         };
27         break;
28     case WM_MENUSELECT:
29         hSharedMenu = GetProp(hwndFrame, "SharedMenu");
30         CombinedMenu = hSharedMenu -> menu;
31         count = 0;
32         for (current = 1st menu entry of CombinedMenu; current <= last menu entry
33             of Combined Menu; current++) {
34             if (current -> item_id == hiword (lparam))
35                 found = true;
36             count = count + 1;
37         }
38         if (found)
39             GetFocus(Old_Focus);
40             SetFocus(hwndFrame);
41             descr = hSharedMenu -> descriptor;
42             saveMenuRoutine = NULL;
43             for (d = 0; d <= sizeof (descr); d++) {
44                 if (d = 0) and (count <= descr [d].count)
45                     saveMenuRoutine = descr[d].function;
46                 else if (count > descr[d - 1].count) and (count <= descr [d].count)
47                     saveMenuRoutine = descr[d].function;
48             }
49             break;
50     default:
51         Old_Filter = GetProp(hwndFrame, "OldFilter");
52         call_Old_Filter (hwndFrame, message, wParam, lParam);
53         break;
54 }
55 return ( );
!

```

コードテーブル5は、オブジェクトリンク及び埋め込みAPIにより与えられる特殊なメッセージハンドラーの具現化を示している。ライン3ないし54は、受け取ったメッセージの形式に基づくケースステートメントを具現化する。ライン28ないし49は、ハンドラーがその基礎となるウィンドウシステムからメニューコマンドメッセージを受け取ったときにメッセージを適切に送れるようにする初期化を与える。ライン4ないし27及びライン50ないし53は、メニュー

コマンドメッセージが受け取られたときに基本的なルートメカニズムを与える。

ライン29において、共用メニューデータ構造に対するハンドルは、コンテナアプリケーションのフレームウインドウの特性から検索される。ライン32ないし37において、ハンドラーは、複合メニューバーのエントリーをウォークし、その入力メッセージで受け取られたメニューとの一致を試みる。このプロセスにおいて、遭遇するメニューの数を追跡する（ライン36）。ループを出たときにメニューが見つかった場合には、変数カウントは、左から始めて選択されたメニューの数を表す。このカウントは、次いで、ライン43ないし48に含まれたループに使用され、メニューがどの記述子エレメントに属するかを決定する。より詳細には、記述子の各インデックスに記憶された値がチェックされ、そしてメニューの数がその値以下であって且つ左側のインデックスにおいて記述子に記憶された値より大きい場合には、正しい記述子エレメントが位置決めされる。正しいエレメントが分かると、ハンドラーは、コンテナアプリケーションファンクションを呼び出さねばならないかサーバアプリケーションファンクションを呼び出さねばならないかに対応する指示子を検索する。この指示子は、次いで、ライン45及び47において変数 `saveMenuRoutine` にセーブされる。メニューのニューモニックス (mnemonics) (システムメニューキーシーケンスを含む) を適切に処理するためのハンドラーとして、このハンドラーは、メニューコマンドメッセージに応答するときに入力フォーカスをフレームウインドウにセットする。

メニューのニューモニックスは、マウスを使用するのではなく、キーボードを使用してメニューをアクセスする方法を与える「Alt. -, F. N」のようなキーシーケンスである。通常は、メニュー項目は、例えば、ウインドウズ3.1においてアンダーラインを付したその独特のニューモニックスとして指定された1つの文字を有する。メニュー項目をアクセスするために、その独特のニューモニックスがその収容するメニューのニューモニックスに添付される。全キーシーケンスの前にはシステムキーがあり、これはユーザがメニューニューモニックスをタイプ入力したい旨をシステムに知らせる。メニューニューモニックスは、イン

・ プレース対話がその基礎となる幾つかのウィンドウシステムで具現化されるときに特殊な問題を課する。より詳細には、変更されない条件のもとで、コンテナオブジェクトがその位置でアクチベートされたときに、システムキー（ウィンドウ3.1における「ALT」キー）及び他のナビゲーションキーを除く全てのキーボード入力がオブジェクトルートウィンドウに対するウィンドウ手順に送られる（図13の項目1309を参照されたい）。というのは、オブジェクトルートウィンドウには、オブジェクトがアクチベートされたときに入力フォーカスが与えられるからである。しかしながら、その基礎となるウィンドウシステムはシステムキーをコンテナアプリケーションフレームウィンドウのウィンドウ手順に送る。というのは、このキーは特殊に手渡しされるからである。従って、コンテナアプリケーションは、システムキーシーケンスの選択されたメニュー項目に対応するキー入力を受け取ることではない。例えば、メニューニューモニックスのキーシーケンスがALT, -, m（「move」に対し）である場合には、「ALT」キーの押圧がコンテナアプリケーションへ送られそして「-」及び「m」キーの押圧がサーバアプリケーションへ送られる。この問題を解消するために、入力フォーカスはコンテナアプリケーションフレームウィンドウに一時的にセットされ、一方、フレームウィンドウ手順はメニュー事象を処理する。この解決策の一例がライン5、39及び40に示されている。

ライン4ないし14において、ハンドラーがメニューコマンドを受け取るときに、ハンドラーは、先ず、通常のプロセスを続けられるように入力フォーカスを回復させる。次いで、ハンドラーは、SaveMenuRoutineにセーブされた指示子をチェックする。これが、メニューがサーバアプリケーションに属することを指示する場合には、ハンドラーはオブジェクトウィンドウのウィンドウ手順を呼び出し、さもなくば、元のコンテナアプリケーションウィンドウ手順を呼び出す。より詳細には、ライン7において、ハンドラーは、コンテナアプリケーションのフレームウィンドウの特性として記憶されたオブジェクトのウィンドウハンドルを得る。ライン8において、ハンドラーは、オブジェクトウィ

ンドウにメッセージを非同期で発信し、元のメッセージ及び元のパラメータを送

る。ライン11において、指示子がサーバアプリケーションを特定しない場合には、ハンドラーは、コンテナアプリケーションのフレームウインドウの元のウインドウ手順を検索し、そしてライン12において、この手順を、指定されたメッセージ及びパラメータで呼び出す。コンテナアプリケーションのフレームウインドウの元のウインドウ手順は、新たなウインドウ手順においてインストールされたObjectSetMenuDescriptorへの呼び出しにおいてセーブされる。

ライン15ないし27において、ハンドラーは、他のメニュー関連メッセージを受け取ると、ライン6ないし13で与えられたのと同様にコンテナ又はオブジェクトアプリケーションへメッセージを適宜発送する。ライン50ないし53において、他の種類のメッセージが受け取られた場合には、古いウインドウ手順が検索され、メッセージ及びパラメータがそこに送られる。

5. 3 その位置でのデアクチベーション手順

ユーザが、その位置でアクチベートされるオブジェクト以外のエリアを選択すると、オブジェクトがデアクチベートされる。一般に、この振る舞いは、ユーザがコンテナアプリケーションのウインドウか又はMDIコンテナアプリケーションの場合には異なる文書ウインドウの別のエリアにおいてマウスボタンをクリックするときに生じる。それ故、一般的に述べると、イン・プレーズ対話APIのデアクチベーション方法は、マウスボタン事象を処理するためにコンテナアプリケーションによって呼び出されるファンクションから呼び出される。

図31は、Process_Mouse_LButtonDownファンクションの具現化を示すフローチャートである。このファンクションは、左のボタンダウンメッセージの受信によって信号される入力事象を処理する。当業者であれば、このようなファンクションは、いずれかの種類の入力事象を受信した際に呼び出すことができそして他の事象は選択及び選択解除に使用できることが明らかであろう。ステップ1301では、このファンクションは、コンテナアプリケーションがアクチベートしようとしていることを指示するフラグをセットする。このフラグは、ネスト状デアクチベーションの呼び出しのシーケンス中に使用さ

れ、ネスト状アクチベーションの場合に正しいユーザインターフェイスリソースが表示されるように確保する。ステップ3102において、このファンクションは、フラグACTIVATED_SOMEONEをチェックして、オブジェクトがその位置でアクチベートされたかどうかを判断し、もしそうであれば、ステップ3104へ続き、さもなければ、ステップ3103へ続く。ステップ3103では、このファンクションは、その通常の左ボタンダウン事象処理を実行し、そして復帰となる。ステップ3104では、このファンクションは、現在アクチベートされたオブジェクトに対しIOLEInPlaceObjectインターフェイスを検索する。ステップ3105において、このファンクションは、オブジェクトのIOLEInPlaceObject::InPlaceUIDeactivate方法呼び出す。このファンクションは、次いで、ステップ3103へ続く。

図21に示すように、現在アクチベートされたオブジェクトのIOLEInPlaceObject::InPlaceUIDeactivate方法は、次いで、そのIOLEInPlaceParent::OnUIDeactivate方法呼び出して、コンテナアプリケーションがそのユーザインターフェイスリソースをインストールできるようにする。このデアクチベーションは、最上位レベルのコンテナはアクチベートしようとしているオブジェクトのコンテナのいずれかに到達するまで上方にネスト構成になる。(図19を参照されたい)。例えば、図4を参照すると、埋め込まれたスプレッドシートオブジェクト405内に示された埋め込まれたチャートオブジェクト409が現在アクチベートされたオブジェクトでありそしてユーザがスプレッドシートオブジェクト405を選択してこれをその位置でアクチベートする場合は、チャートのIOLEInPlaceObject::InPlaceUIDeactivate方法が呼び出され、これは、次いでスプレッドシートのIOLEInPlaceObject::OnUIDeactivate方法呼び出す。この後者の方法は図4に示すようにスプレッドシートオブジェクトに対するユーザインターフェイスをインストールする。一方、ユーザが複合文書内のどこかでクリックした場合には、スプレッドシートのIOLEInPlaceObject::OnUID

`reactivate`方法は、複合文書の`I O L E I n P l a c e O b j e c t : : O n U I D e a c t i v a t e`方法を呼び出す。この後者の方法は、ワードプロセスユーザインターフェイスをインストールする。というのは、このオブジェクトは最上位レベルのコンテナオブジェクトだからである。

コンテナアプリケーションは、表示されたコンテナオブジェクトを垂直又は水平にスクロールするためのスクロールバーを表示することができる。これらスクロールバーは、コンテナウィンドウの一部分である。コンテナオブジェクトがアクチベートされたコンテナオブジェクトを有する場合には、ユーザがコンテナオブジェクトの外側のエリアでクリックしたときに、コンテナオブジェクトはデアクチベートされる。好ましい実施例では、コンテナアプリケーションは、スクロールバーメッセージを受け取る際に、コンテナオブジェクトをデアクチベートしない。むしろ、コンテナアプリケーションは、スクロールを行い、入力フォーカスがコンテナオブジェクトと共に留まるよう確保する。

5. 4 コンテナアプリケーションを閉じる

コンテナアプリケーションが、既にアクチベートされたオブジェクトを再アクチベートする開放 (undo) オペレーションをもはや実行できなくなった後であって、且つコンテナアプリケーションがユーザによって閉じられる前のある時間に、コンテナアプリケーションは、既にアクチベートされたオブジェクトに関連したユーザインターフェイスリソースを永久的に割り当て解除する。これらのリソースを割り当て解除するために、コンテナアプリケーションは、既にアクチベートされたオブジェクトに関連した`I O L E I n P l a c e O b j e c t : : I n P l a c e D e a c t i v a t e`方法を呼び出す。この方法は、次いで、共用メニューデータ構造と、複合メニューバーに関連したメニューとを割り当て解除する。(図20及びそれに関連した説明を参照されたい。)

5. 5 モードレスダイアログとの対話

その位置でアクチベートされたオブジェクトであってそのサーバアプリケーションがモードレスダイアログを表示しているようなオブジェクトとユーザが対話するときに、ユーザがそれ自身のモードダイアログを表すコンテナアプリケーションメニューからメニュー項目を選択したい場合には、コンテナアプリケーション

ションは、サーバアプリケーションのモードレスダイアログを一時的に隠す。モードレスダイアログボックスが隠されるのは、ユーザが同時に表示される2つのダイアログボックスを見て混乱しそしてどのボックスに入力が送られるか分からないからである。というのは、サーバ及びコンテナアプリケーションは、1つのアプリケーションとして現れることを意味するからである。又、モードダイアログは、他のダイアログとの競合を回避するようにプログラムされていない。というのは、慣例的なアプリケーションにおいては、その基礎となるウィンドウシステムが単一のアプリケーション内のモードダイアログの外側の入力を禁止するからである。イン・プレーズ対話を使用する場合には、2つのアプリケーションが共働して1つに見えるので、このような競合は自動的に回避される。それ故、アプリケーションは、モードダイアログとモードレスダイアログとの間の競合を回避するように共働しなければならない。例えば、ユーザがスプレッドシートアプリケーションのEditメニューにおける「Find...」メニュー項目を選択し、サーバアプリケーションによってモードレスダイアログが表示されると仮定する。ここで、ユーザが複合文書の一部分をプリントアウトしようとし、従って、ユーザが、ワードプロセス（コンテナ）アプリケーションに属するFileメニュー上の「Print...」メニュー項目を選択すると仮定する。ワードプロセスアプリケーションは、両方のダイアログが同時に表示されないのが好ましいので、「Find...」ダイアログを隠す。これを行うために、ワードプロセスアプリケーションは、スプレッドシートアプリケーションのIOLEInPlaceObject::EnableModeless方法呼び出して、モードレスダイアログを隠すように要求する。次いで、コンテナアプリケーションが「Print...」ダイアログの処理を終了した後に、EnableModeless方法呼び出して、モードレスダイアログを再表示する。

サーバアプリケーションがコンテナアプリケーションのモードレスダイアログを隠す必要がある場合にも同様の状態が生じ得る。この場合には、IOLEInPlaceFrame::EnableModeless方法が使用される。

5. 6 アクセラレータキーの組合せの取り扱い

本発明の好ましい実施例においては、その基礎となるウィンドウシステムは、

アクセラレータキー組合せと称する概念をサポートし、ユーザがキーボードショートカットを介してメニューコマンドを呼び出すことができるようにする。アクセラレータキー組合せは、特定のメニューコマンドを呼び出すことと同等であるようにアプリケーションによって指定された一連のキーである。例えば、「CTRL」キーを押した後に「N」キーを押すことより成るキーのシーケンスは、「File」メニュー上のメニューコマンド「New」に変換される。典型的なシステムにおいて、アクセラレータキー組合せは、ユーザによって指定でき、アプリケーション内で独特である必要がある。

一般に、アクセラレータキー組合せ（アクセラレータ）は、アプリケーションのメッセージポンプにおいて処理される（図14のステップ1402を参照）。典型的なメッセージポンプは、その基礎となるウインドウシステムを呼び出してアクセラレータ変換テーブルをそこに通し、ウインドウシステムが、アクセラレータがどのメニュー項目コマンドに対応するかを決定するようにさせる。ウインドウシステムは、次いで、それにより得たメニューコマンドを適当なウインドウ手順に送信する。

オブジェクトリンク及び埋め込みイン・ブレース対話APIではアクセラレータに対して問題が生じ得る。まず、サーバアプリケーションはそのコンテナアプリケーションのプロセススペース内でオブジェクトハンドラーとして具現化できるので、コンテナアプリケーションは、サーバアプリケーションがそれ自身のアクセラレータを変換する機会をもつよう確保しなければならない。好ましくは、サーバアプリケーションには、サーバアプリケーションがその位置でアクティベートされたときにあいまいなアプリケーションアクセラレータの処理においてコンテナアプリケーションより高い優先順位が与えられる。又、サーバアプリケーションがそれ自身のプロセススペースにおいて具現化される場合には、それが確認できないアクセラレータをコンテナアプリケーションに通さねばならない。

この問題を解決するために、コンテナアプリケーション及びサーバアプリケーションのメッセージポンプは、アプリケーションアクセラレータを変換する機

会を互いに許すように変更される。コードテーブル6及び7は、サーバアプリケーションが個別のプロセスとして実行されるときに適用できるサーバアプリケーションのメッセージポンプに対する変更を示している。更に、コードテーブル8は、サーバアプリケーションが同じプロセス内でコンテナアプリケーションとして（オブジェクトハンドラーとして）実行されるときに適用できるコンテナアプリケーションのメッセージポンプに対する変更を示している。

Code Table 6

Object's message loop:

```
1  while (GetMessage (&msg, NULL, NULL, NULL)) {  
2      if (not (TranslateAccelerator (hwndObj, hAccel, &msg))) {  
3          if (not (ObjectTranslateAccelerator (&msg, hwndFrame, hAccel Table))) {  
4              TranslateMessage (&msg);  
5              DispatchMessage (&msg);  
6          }  
    }
```

コードテーブル6は、その位置でアクチベートされたオブジェクトのメッセージポンプに対する変更の具現化を示す。これらの変更は、サーバアプリケーションが、コンテナアプリケーションに、サーバアプリケーション（個別のプロセス）が到来メッセージを最終的に破棄する前にアプリケーションアクセラレータを変換する機会を与えることができるようにする。ライン2において、サーバアプリケーションは、それ自身の変換テーブル（変数 `hAccel` に記憶された）を用いてアクセラレータを変換するよう試みる。ライン3において、変換すべきアクセラレータがなかったか、又はサーバアプリケーションの変換テーブルにアクセラレータが見つからなかったことにより、この変換が首尾良くいかなかった場合には、サーバアプリケーションは、特殊なオブジェクトリンク及び埋め込みAPIのファンクション `ObjectTranslateAccelerator`

を呼び出す。このファンクション `ObjectTranslateAccelerator` は、コンテナアプリケーションによってアクセラレータが所望されるかどうか決定し、もしそうならば、遠隔手順通話を介してコンテナアプリケーションにメッセージを送信し、アクセラレータを変換するように要求する。

遠隔手順通話メカニズムは、その同期特性により、発呼者（サーバプロセス）が

更に入力を受け取る前にコンテナアプリケーションがメッセージを処理して復帰するよう確保する。ライン4ないし5において、コンテナアプリケーションがアクセラレータを変換しなかった場合には、サーバアプリケーションが入力メッセージをその通常の形態で処理する（それをフィルタして発送する）。

Code Table 7

```
ObjectTranslateAccelerator(lpmsg, hwndFrame, hAccelFrame){  
1  if (keystroke in lpmsg not found in hAccelFrame)  
2    return (false)  
3  else {  
4    Send RPC message to hwndFrame to invoke  
5    hwndFrame -> TranslateAccelerator  
6    return (value from RPC call)  
  }  
}
```

コードテーブル7は、オブジェクトリンク及び埋め込みAPIのファンクションObjectTranslateAcceleratorの具現化を示している。このファンクションは、サーバアプリケーションがコンテナアプリケーションにアクセラレータを処理する機会を与えることができるようにする。同期メッセージ処理に本来ある落とし穴（不定の待機のような）を回避するために、ObjectTranslateAcceleratorは、まず、コンテナアプリケーションのTranslateAccelerator方法呼び出そうと試みる前にコンテナアプリケーションがアクセラレータに挿入されたかどうか調べるチェックを行う。コンテナアプリケーションのアクセラレータテーブルは、サーバアプリケーションによって通された指定のパラメータである。これ

は、IOLEInPlaceParent::GetWindowContextに対する通話を介してサーバアプリケーションにアクセスできる。コンテナアプリケーションのTranslateAccelerator方法が呼び出された場合には、このファンクションは、コンテナアプリケーションにより返送された値をサーバアプリケーションへ戻し、従って、サーバアプリケーションはメッセージを適切に破棄することができる。

Code Table 8

Container's message loop:

```

1  while (GetMessage (&msg, NULL, NULL, NULL)) {
2      pwhObj = GetActiveObjectHwnd (hwndDoc);
3      translated = false;
4      if ((pwhObj != NULL)) {
5          pipObj = determine the IOLEInPlaceObject interface for pwhObj;
6          if (pipObj -> TranslateAccelerator (&msg))
7              translated = true;
8      }
9      if (not (translated)) {
10         if (not (TranslateMDISysAccel (hwndMDIcontainer, &msg)))
11             if (not (TranslateAccelerator (hwndFrame, hAccel, &msg))) {
12                 TranslateMessage (&msg);
13                 DispatchMessage (&msg);
14             }
15     }

```

コードテーブル8は、イン・ブレース対話をサポートするコンテナアプリケーションのメッセージポンプの典型的な具現化を示す。これらの変更は、コンテナアプリケーションが、サーバアプリケーション（コンテナアプリケーションと同じプロセス内で実行している）に、コンテナアプリケーションが到来メッセージを完全に破棄する前にアプリケーションアクセラレータを変換する機会を与えることができるようにする。ライン2において、コードは、コンテナアプリケーションの文書ウィンドウに関連した現在アクティブなオブジェクトウィンドウハンドルを検索する。ライン4ないし8において、アクティブなオブジェクトウィンドウハンドルがない場合には、コードは、そのオブジェクトウィンド

ウハンドルに対応するIOLEInPlaceObjectTranslateAccelerator方法呼び出して、サーバアプリケーションがアクセラレータキー組合せを変換できるようにする。ライン9ないし11において、サーバアプリケーションがアクセラレータを変換しないか、又はアクティブなオブジェクトがない場合には、コンテナアプリケーションは、それ自身の変換テーブル（変数hAccelに記憶された）を用いてアクセラレータを変換するよう試みる。ライン12ないし13において、変換すべきアクセラレータが確認されない場合には、コンテナアプリケーションは、入力メッセージを通常の形態で処理する（それをフィルタして発送する）。

以上、好ましい実施例について本発明を説明したが、本発明は、この実施例に限定されるものではない。本発明の精神内での変更が当業者に明らかであろう。

本発明の範囲は、以下の請求の範囲によって限定されるものとする。

【図1】

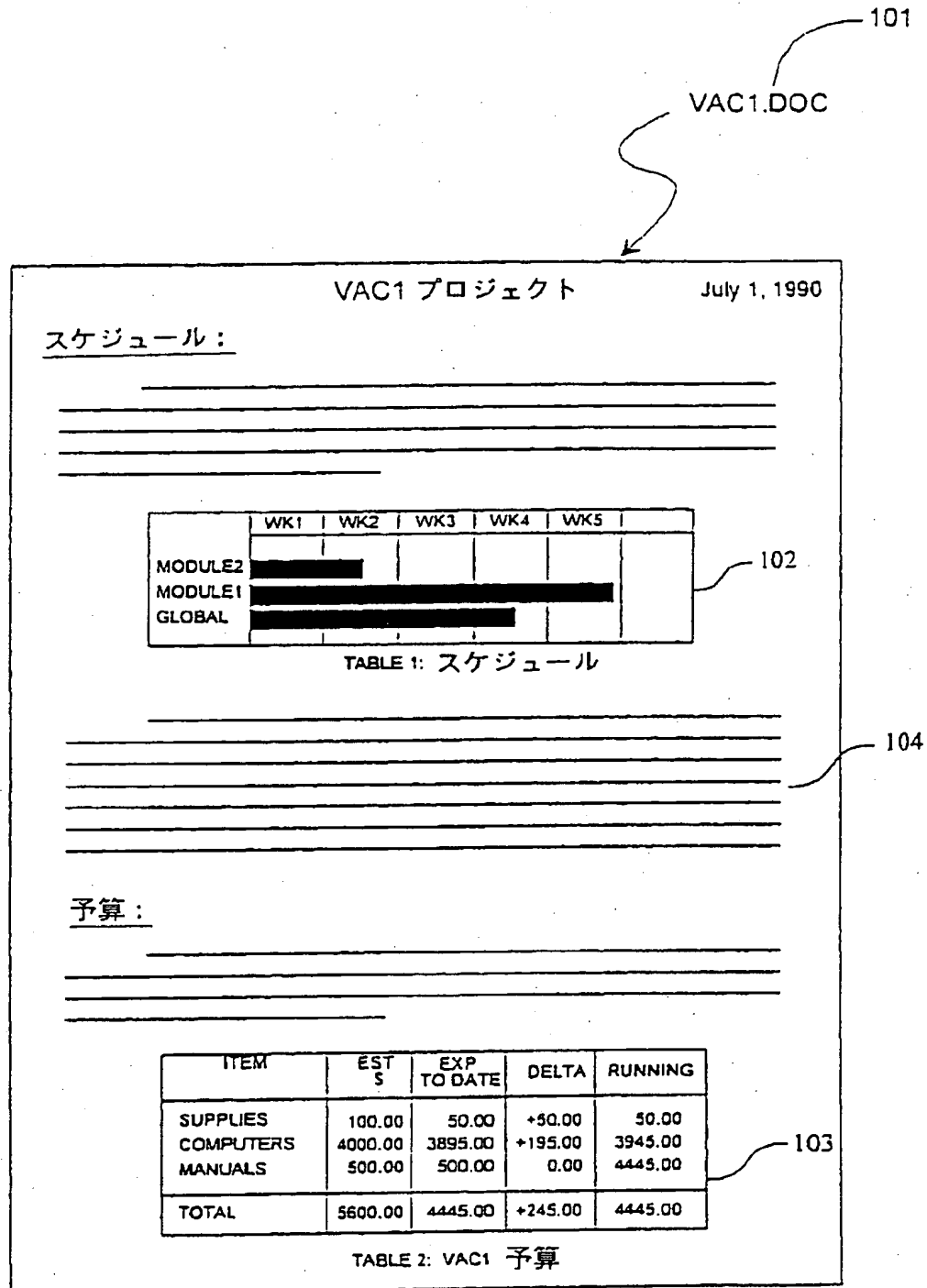


Figure 1

【図2】

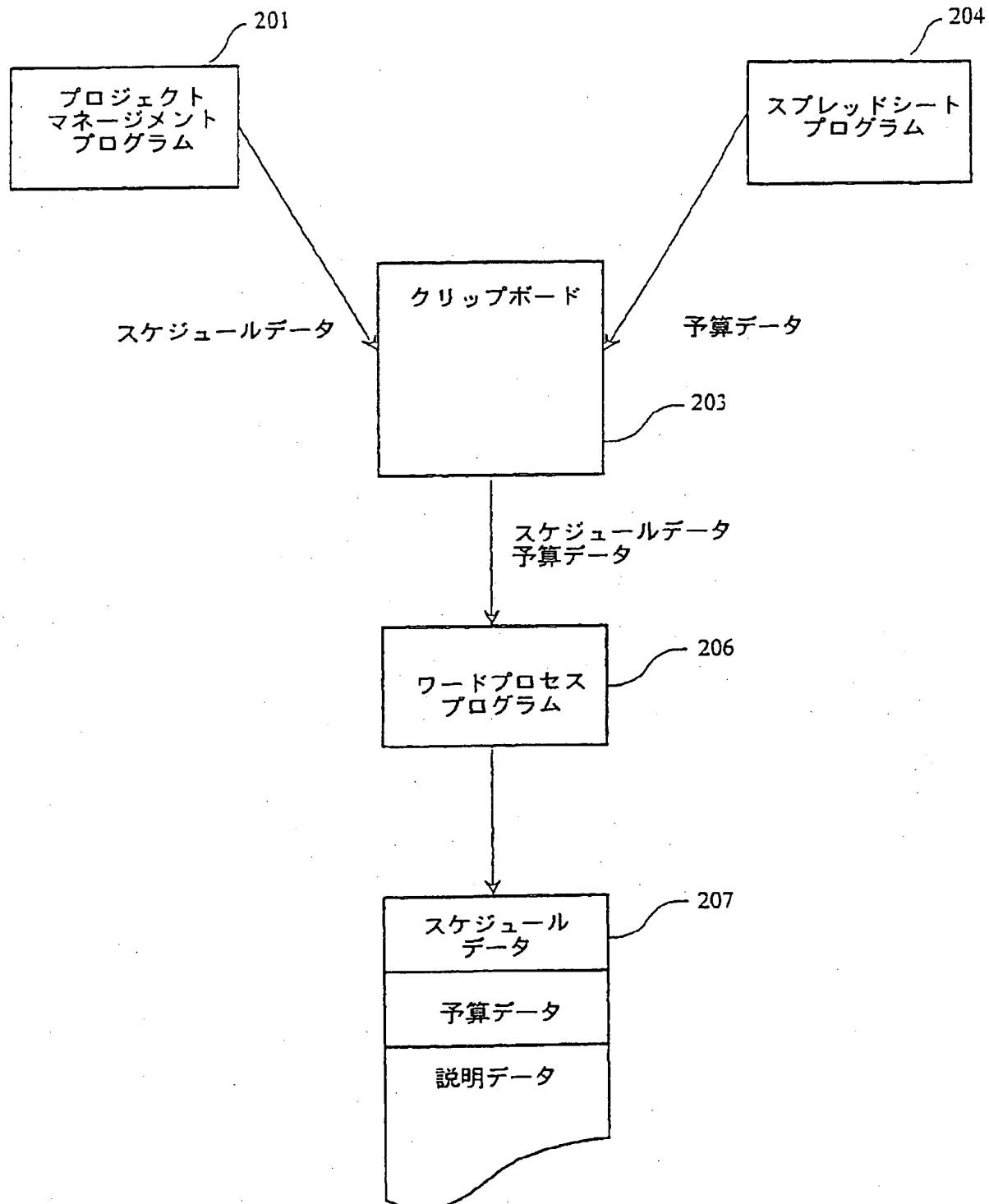


Figure 2

【図3】

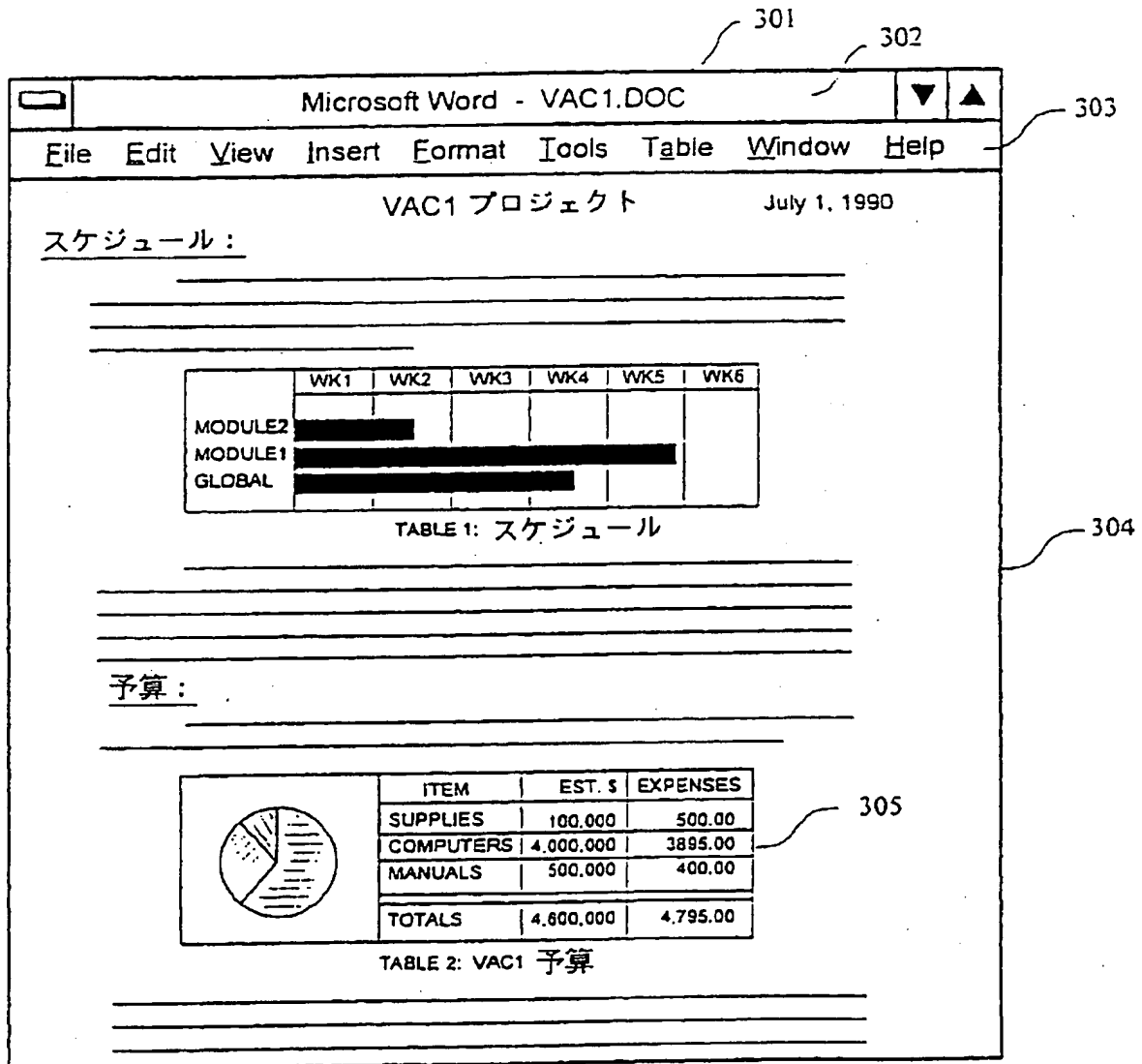


Figure 3

【図4】

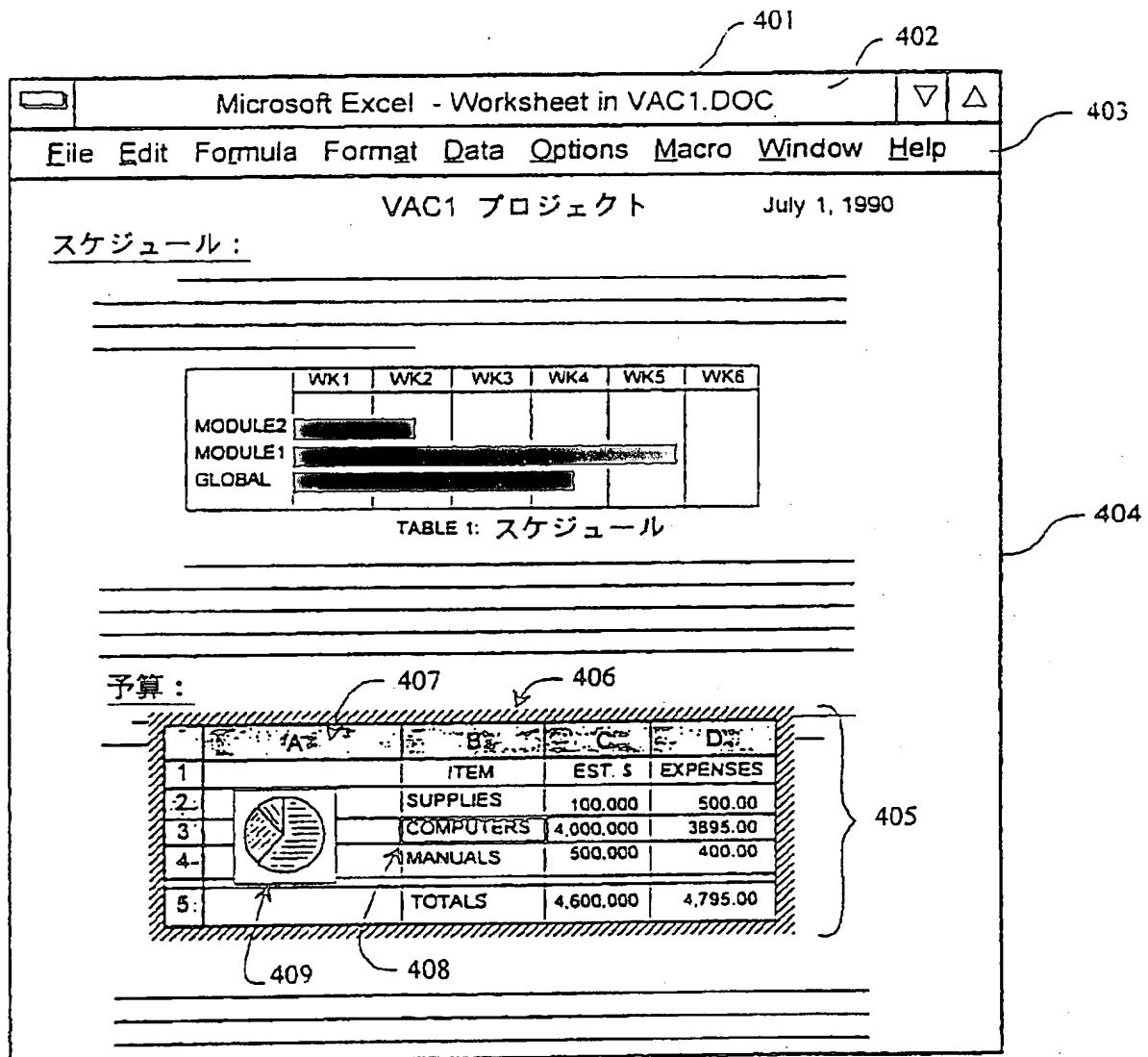


Figure 4

【図5】

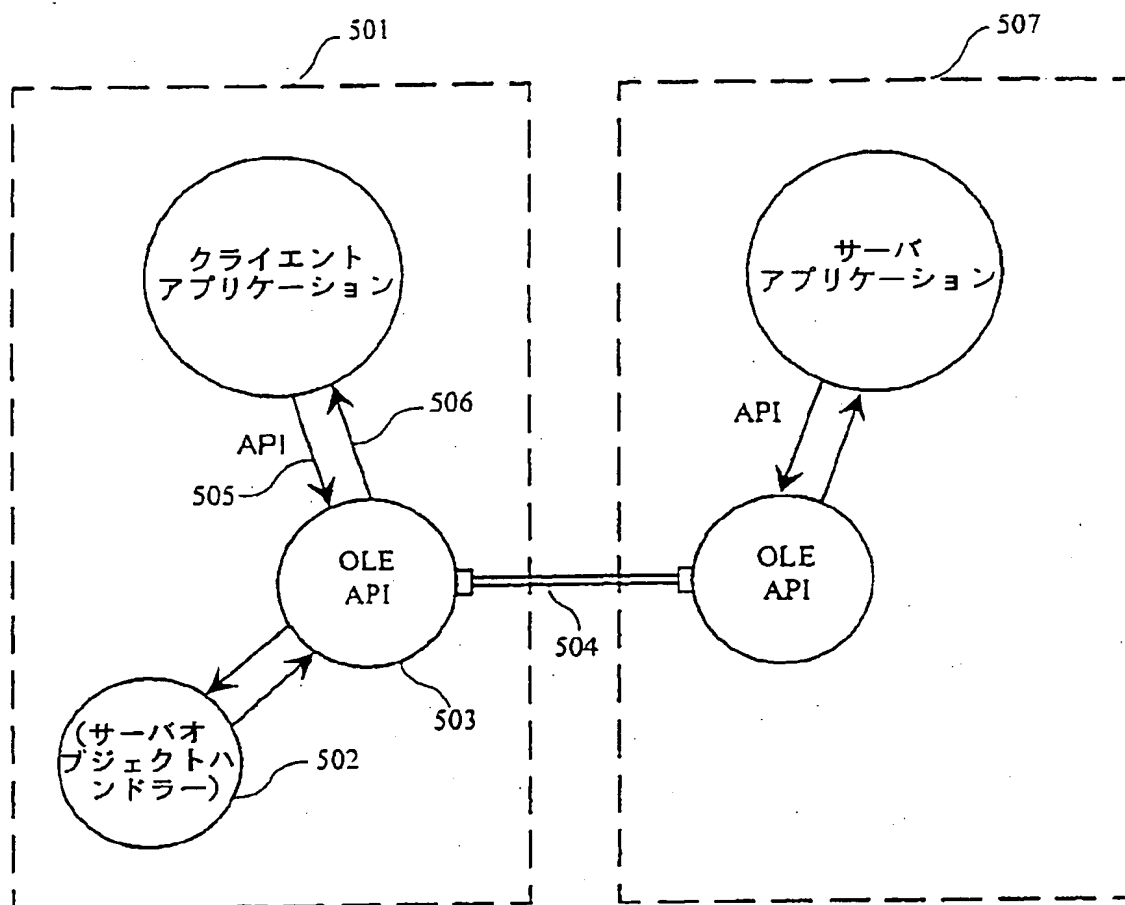


Figure 5

【図6】

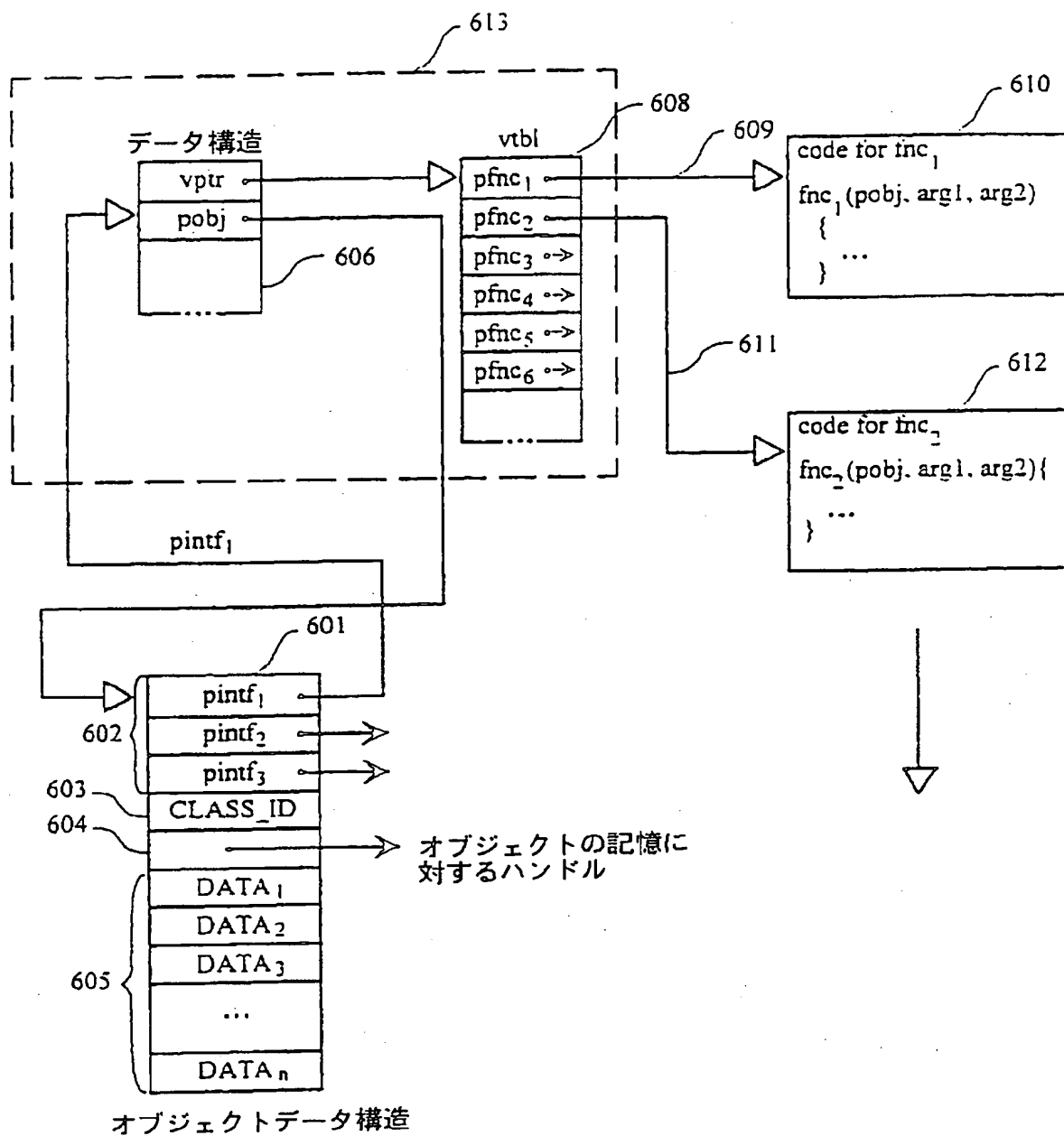


Figure 6

【図7】

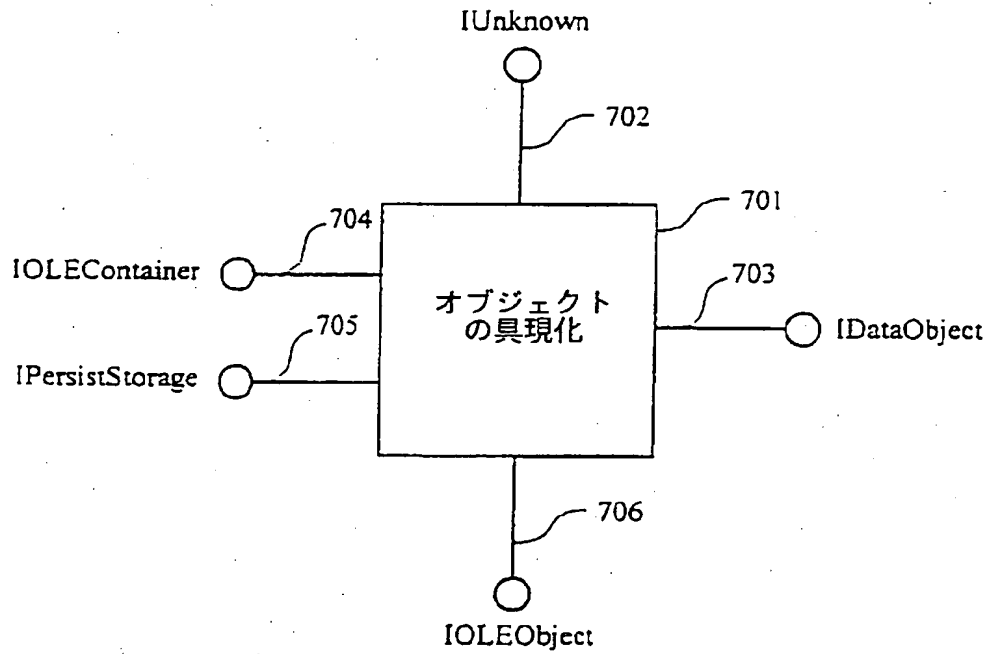


Figure 7

【図8】

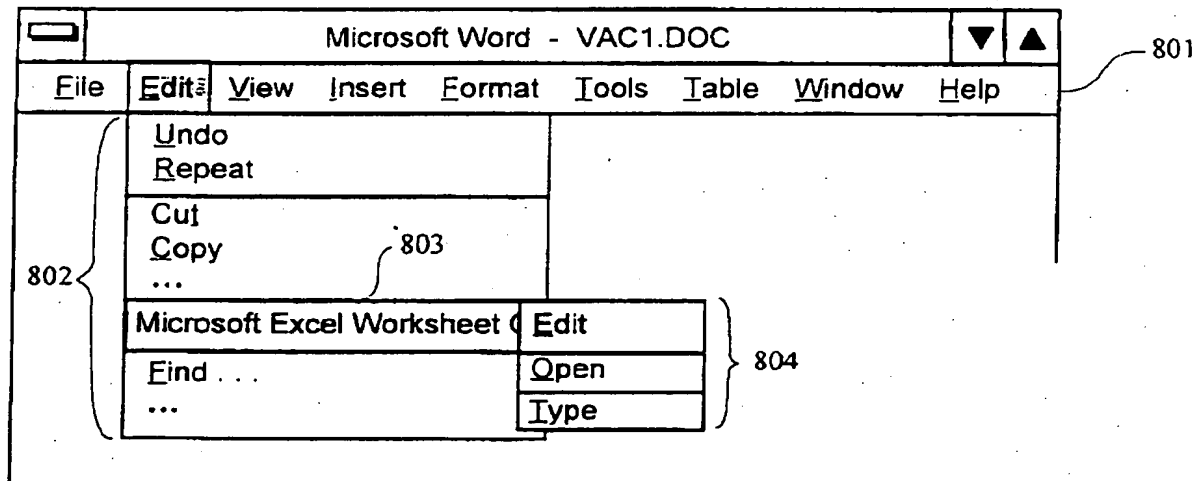
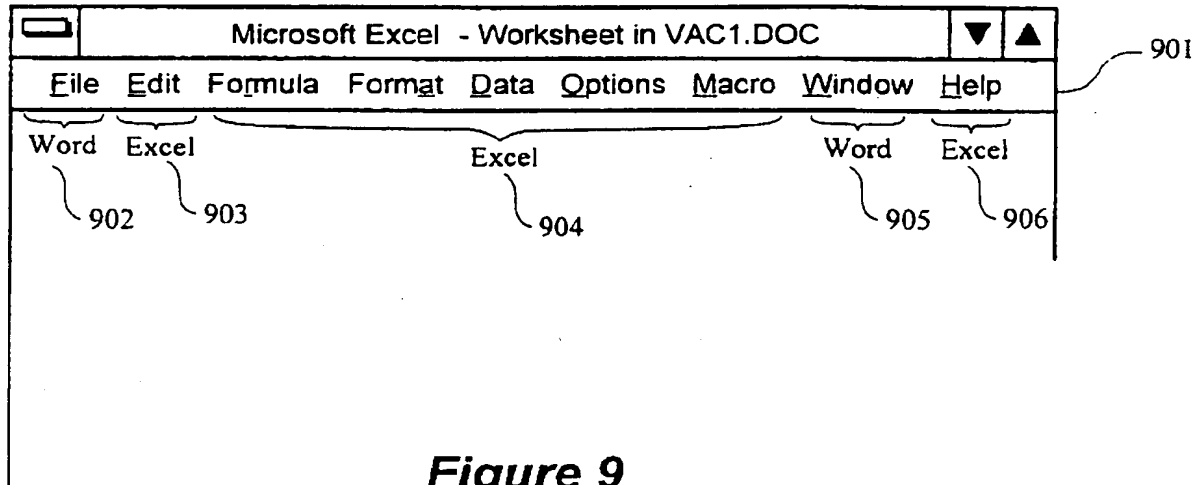


Figure 8

【図9】



【図10】

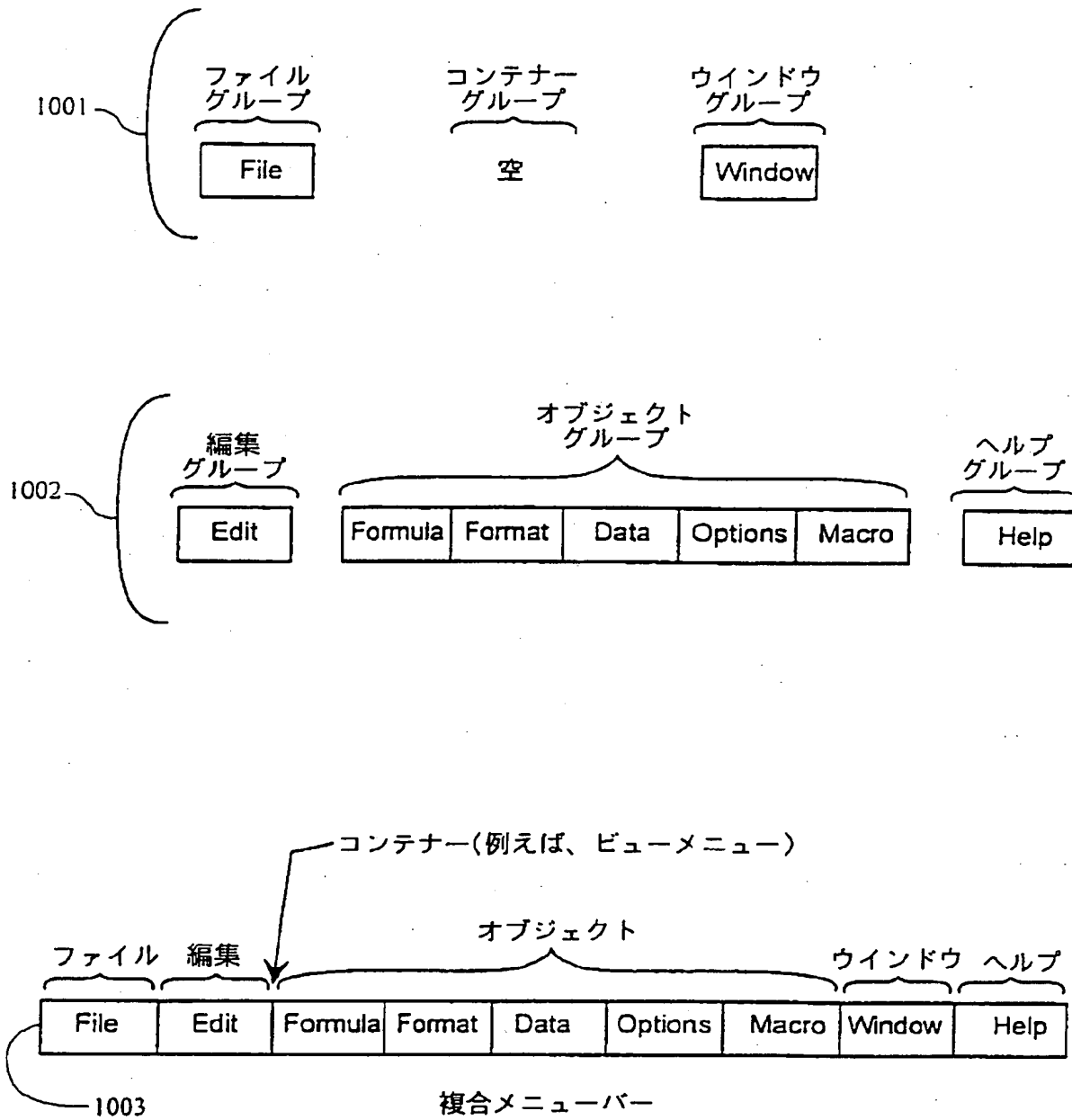


Figure 10

【図11】

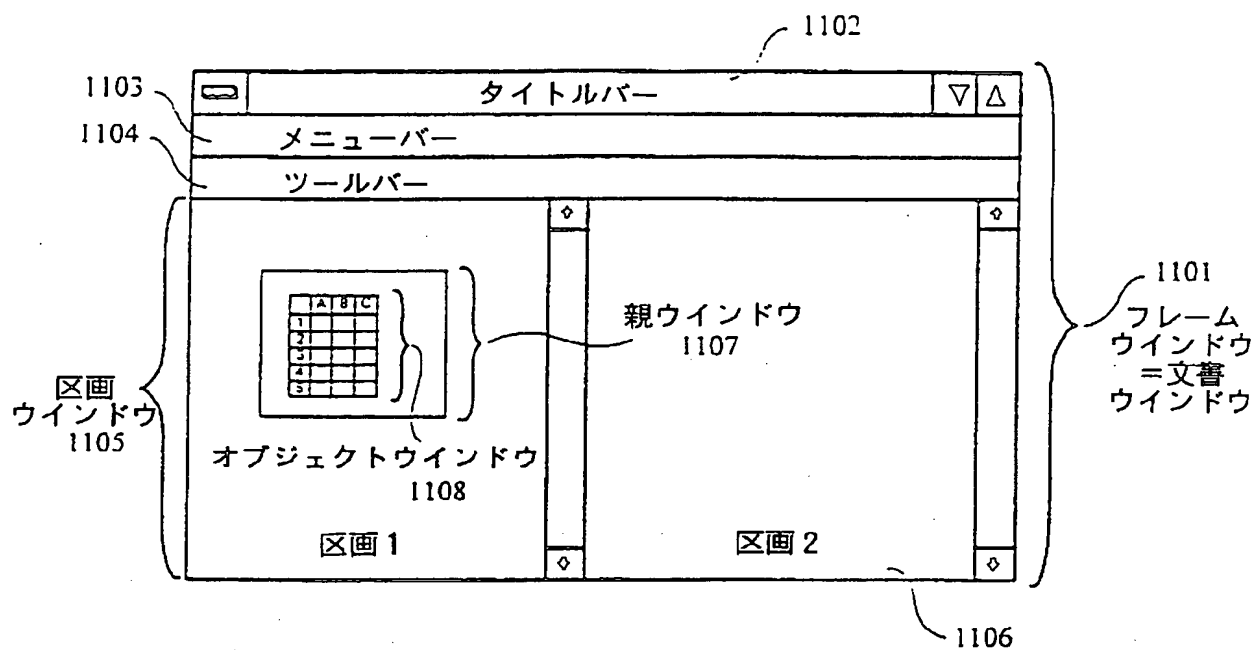


Figure 11

【図12】

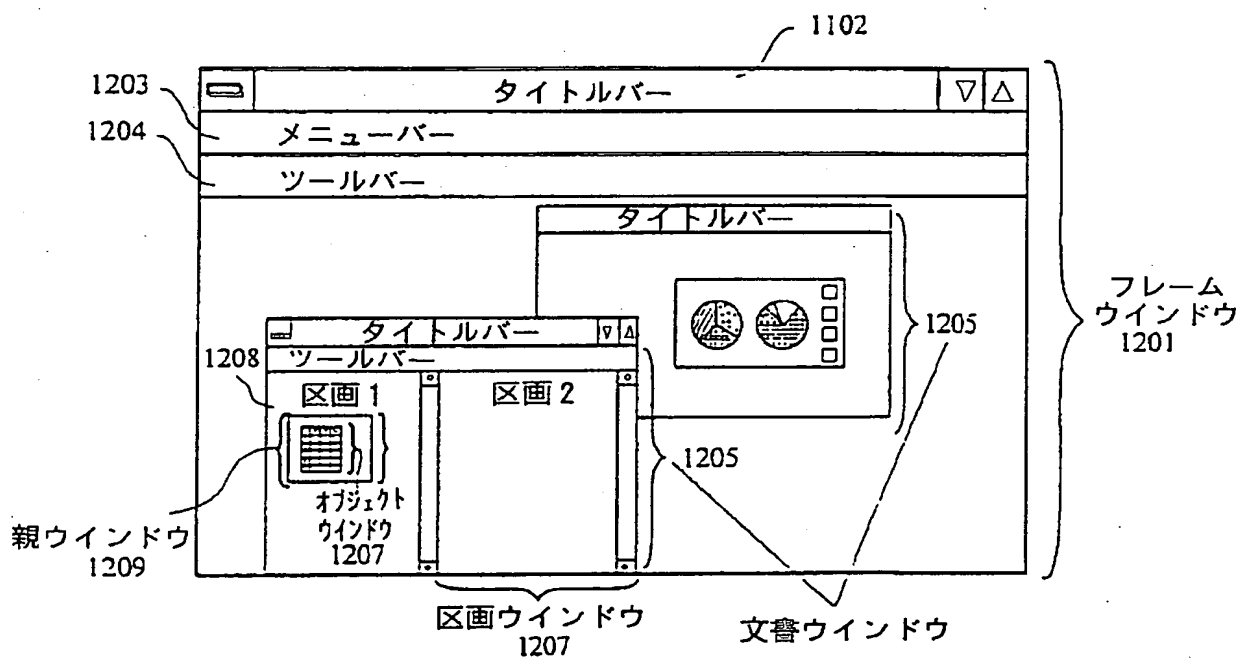


Figure 12

【図13】

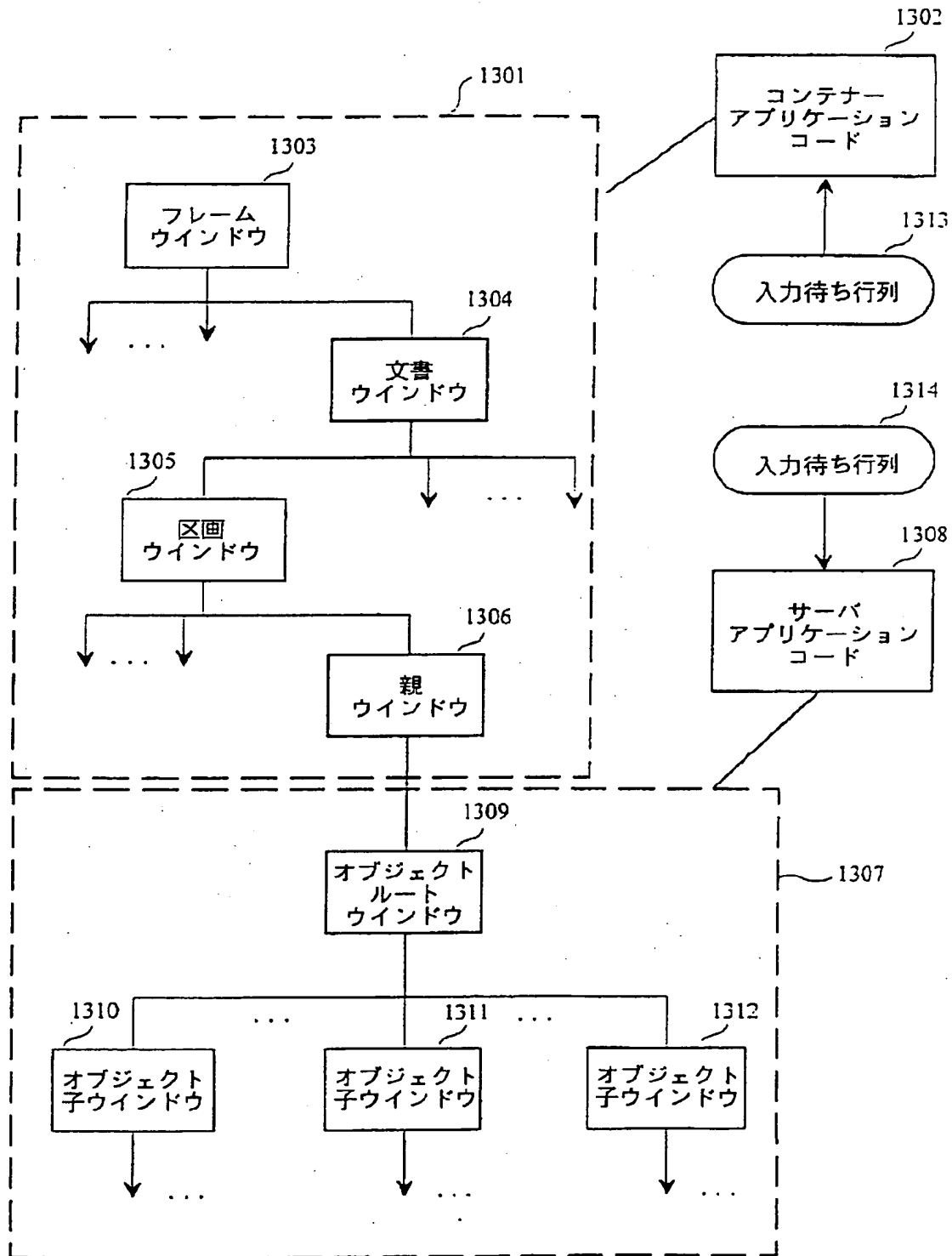


Figure 13

【図14】

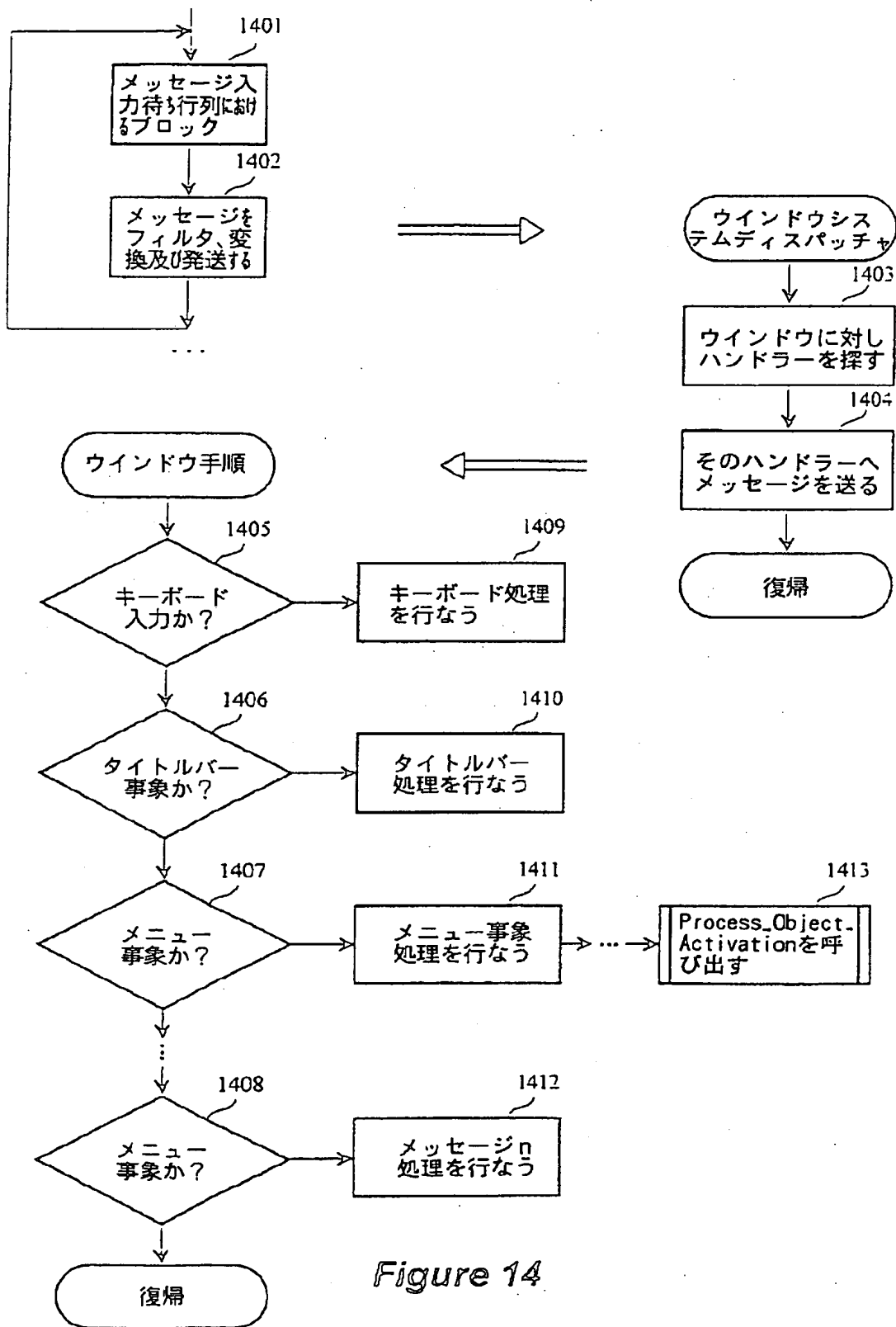


Figure 14

【図14B】

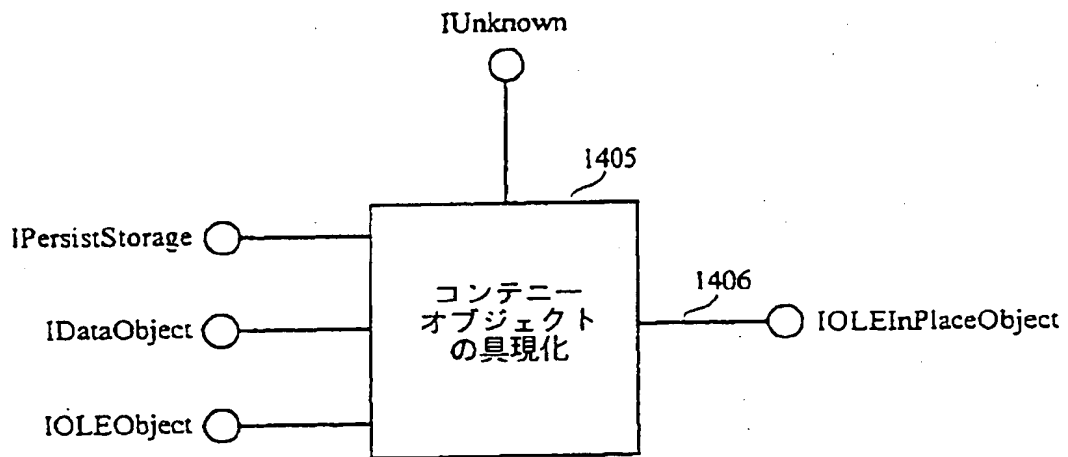
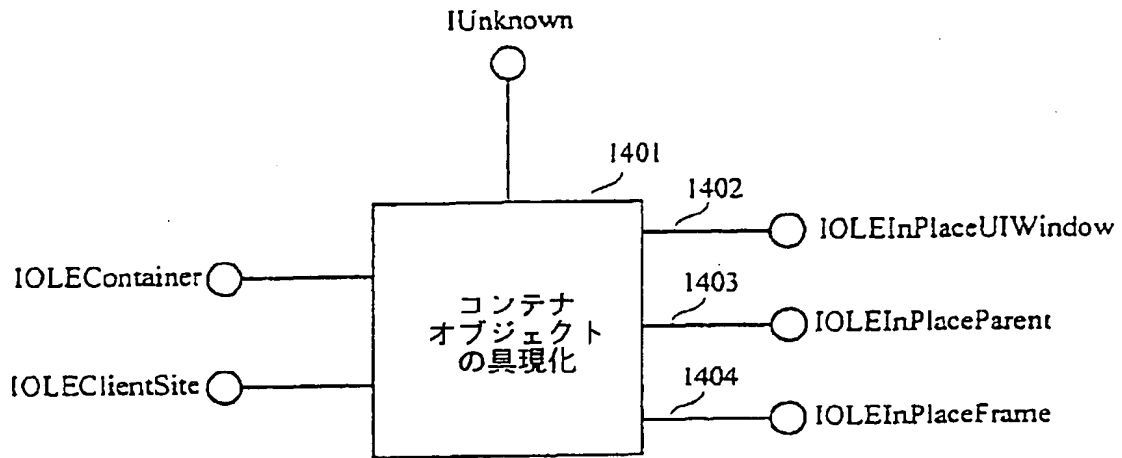


Figure 14B

【図15】

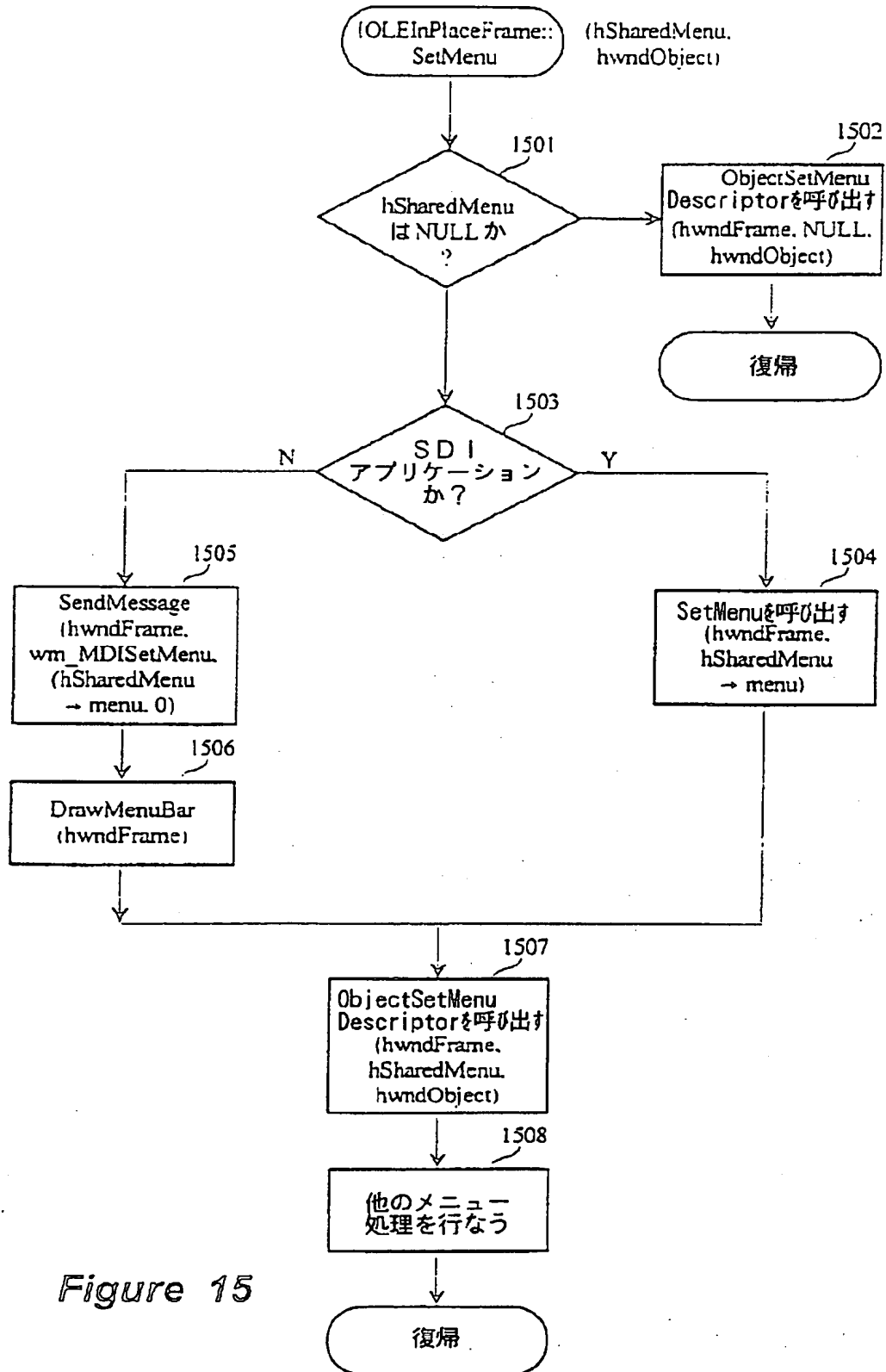


Figure 15

【図16】

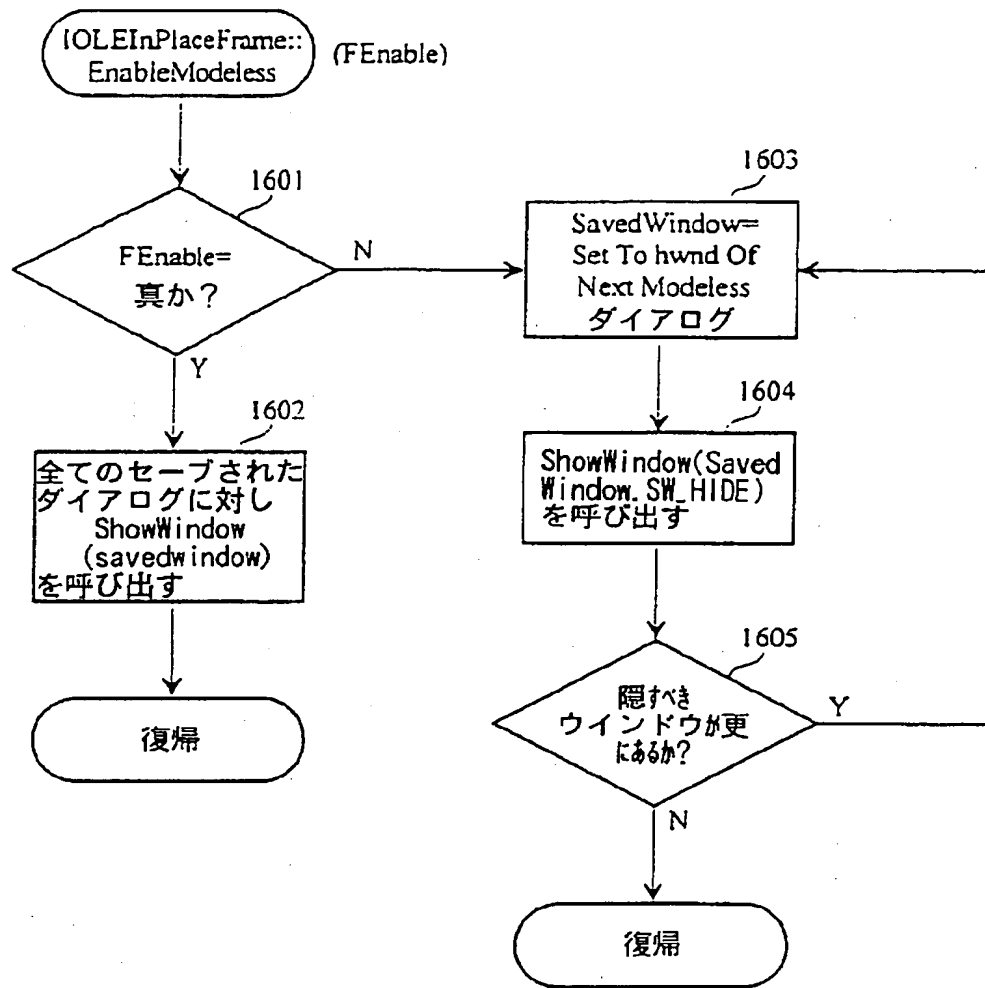
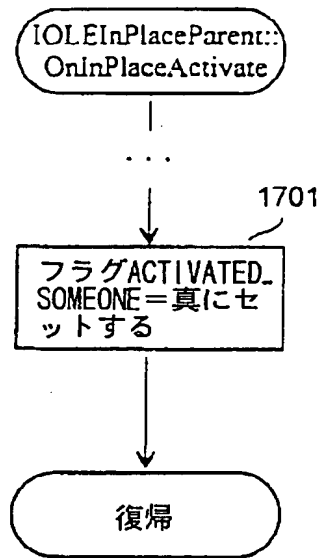


Figure 16

【図17】

**Figure 17**

【図18】

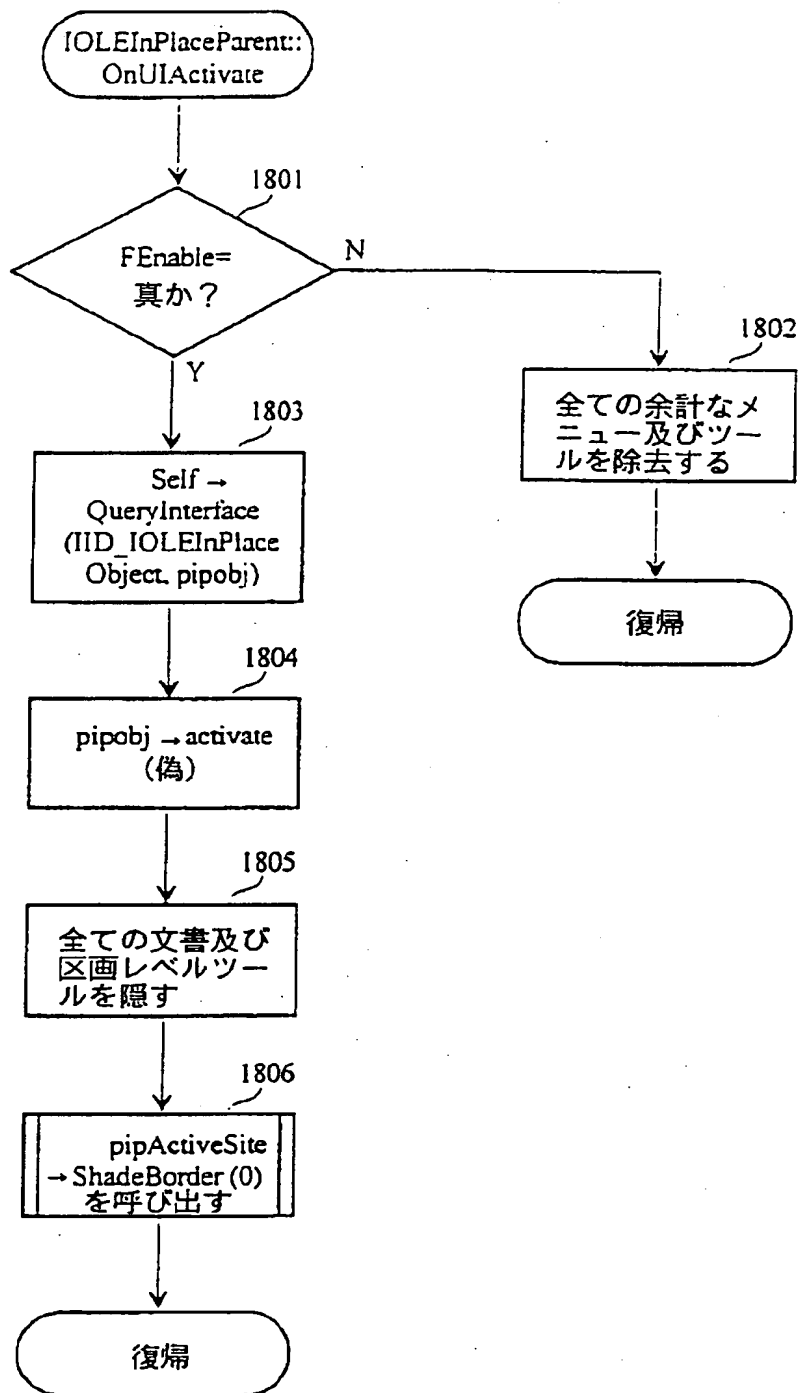


Figure 18

【図19】

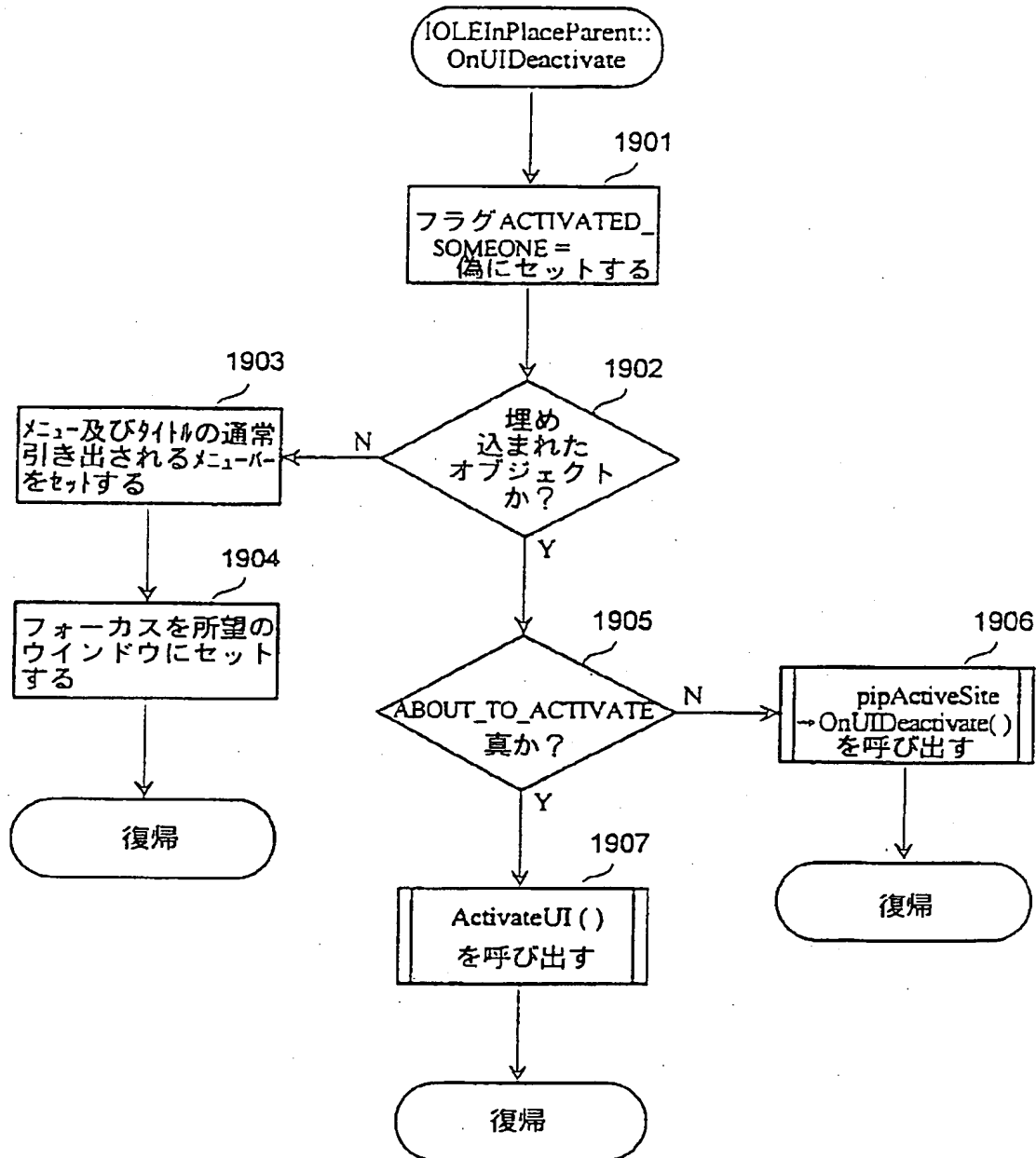


Figure 19

【図20】

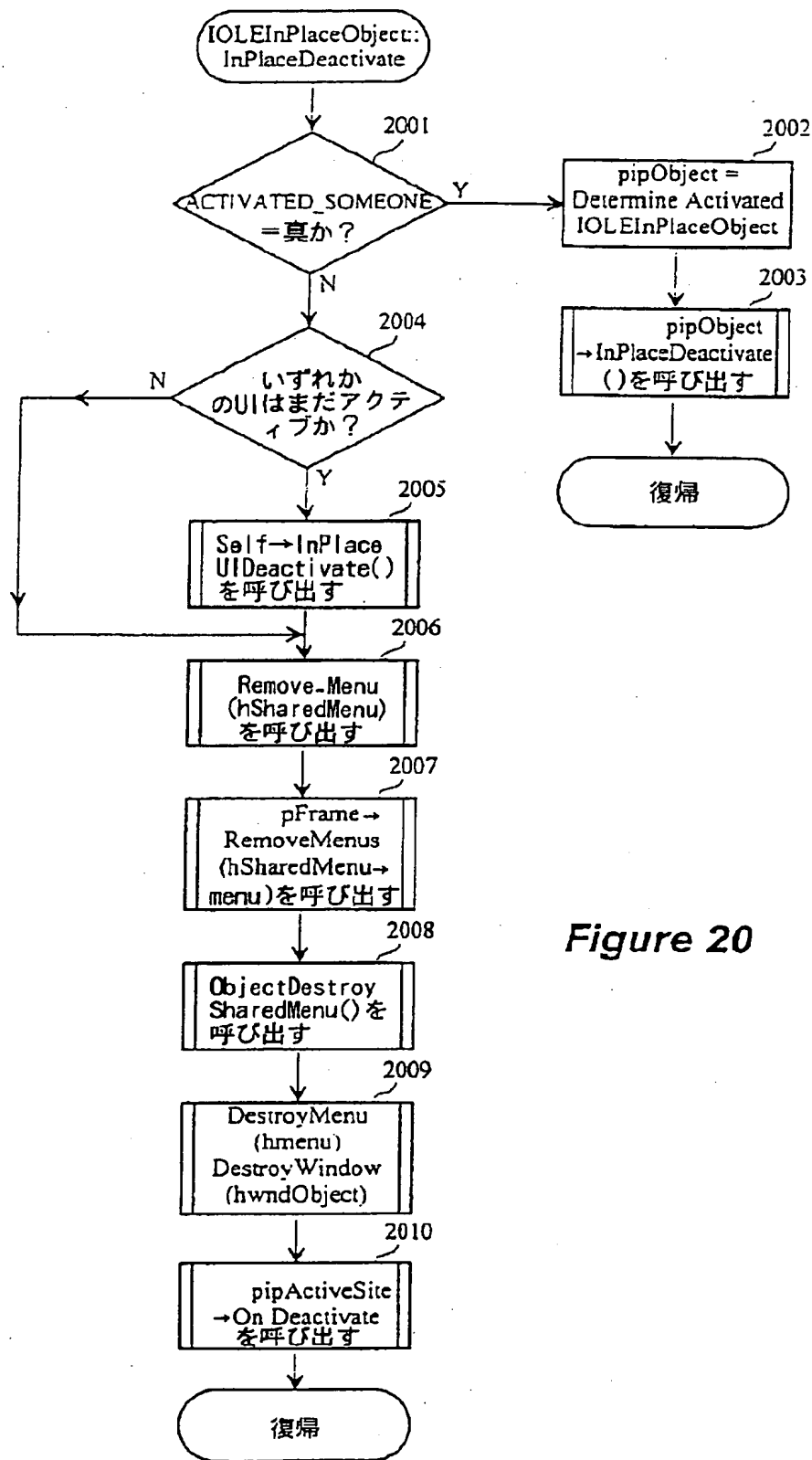


Figure 20

【図21】

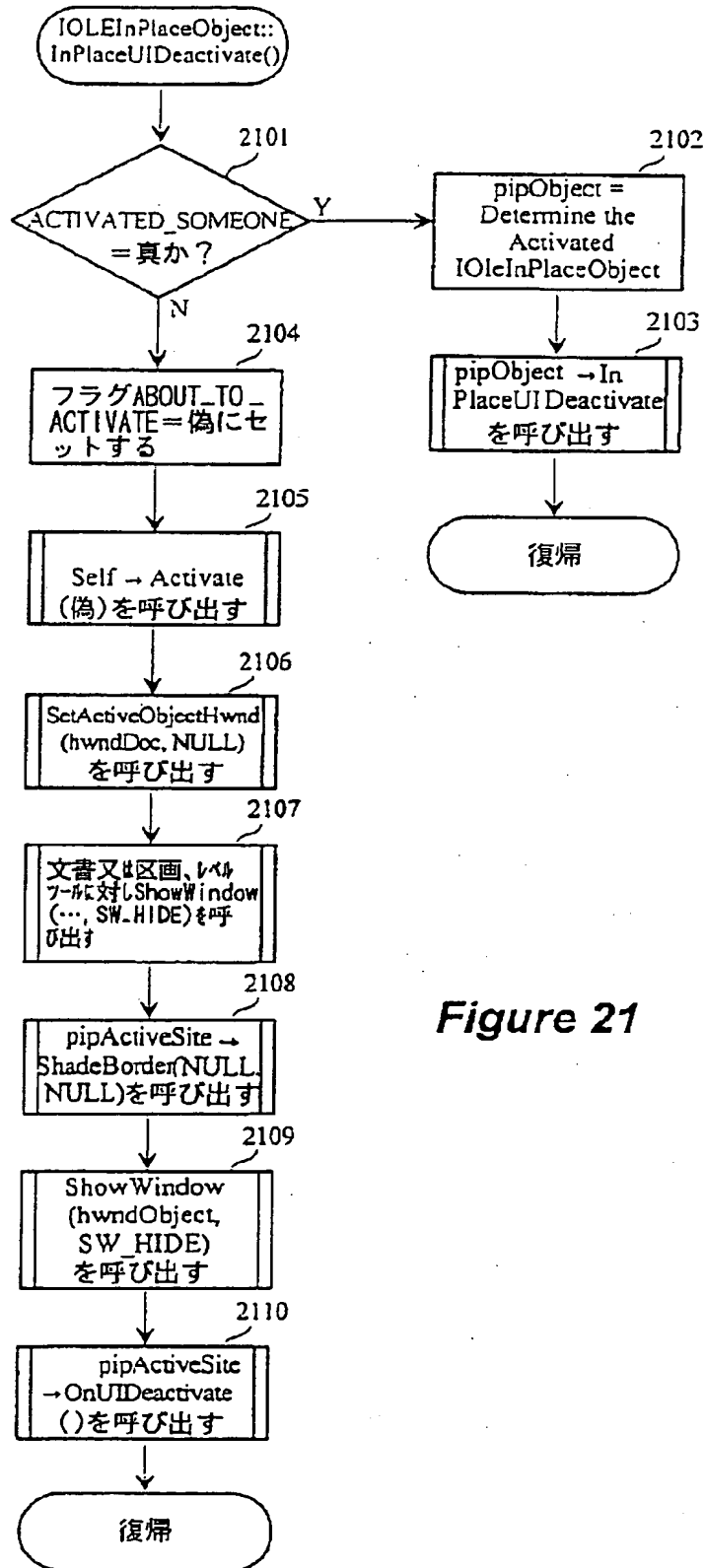


Figure 21

【図22】

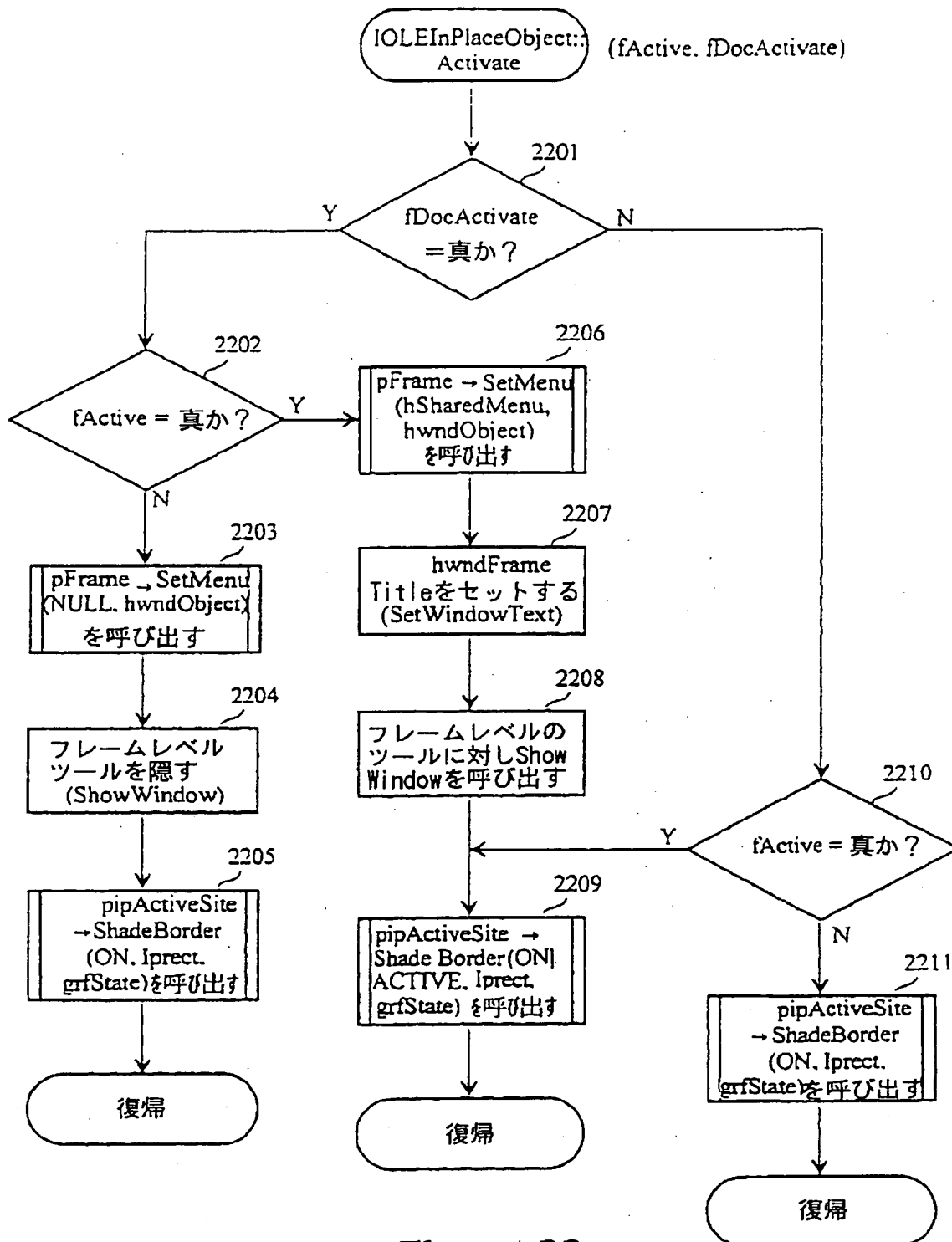


Figure 22

【図23】

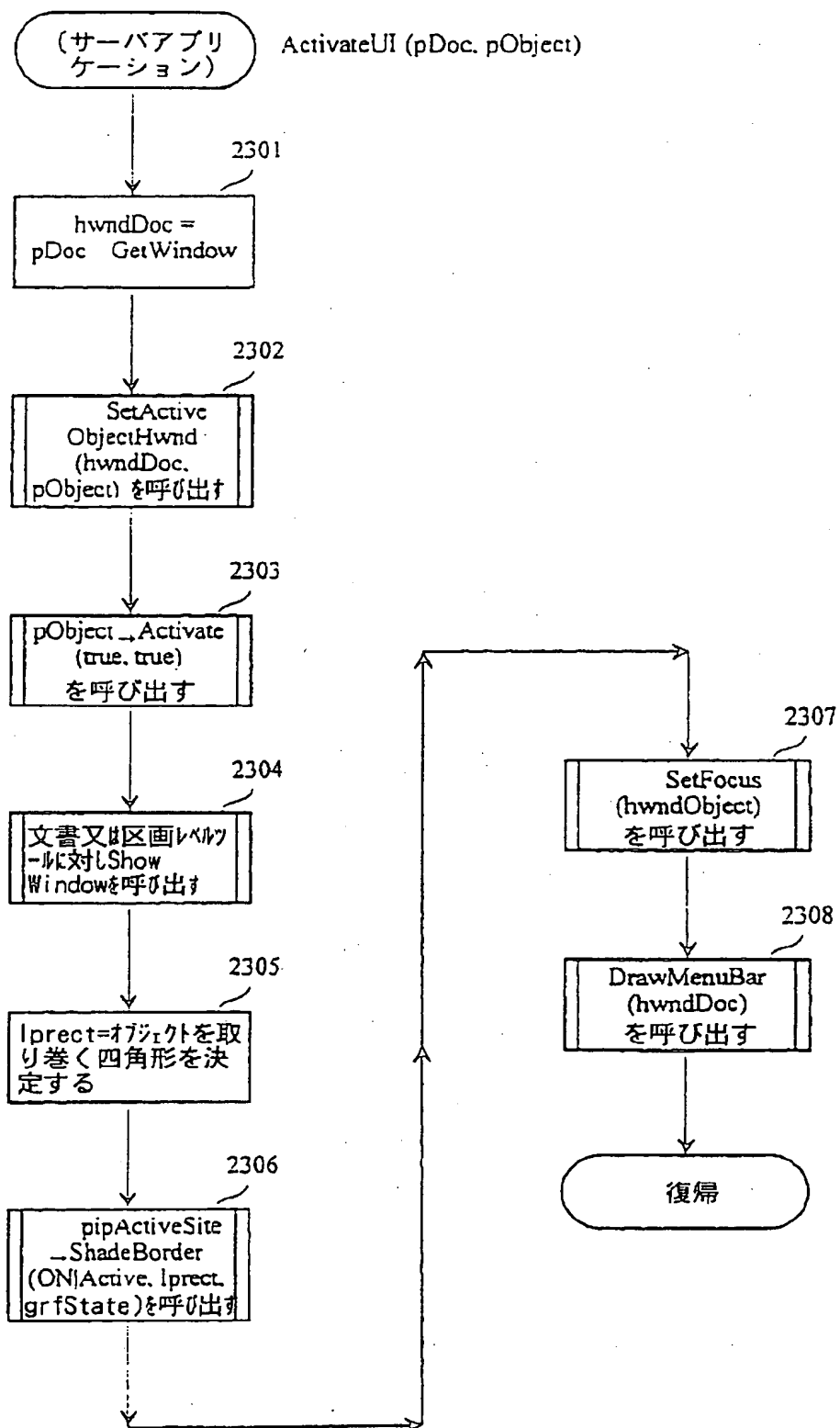


Figure 23

【図 24】

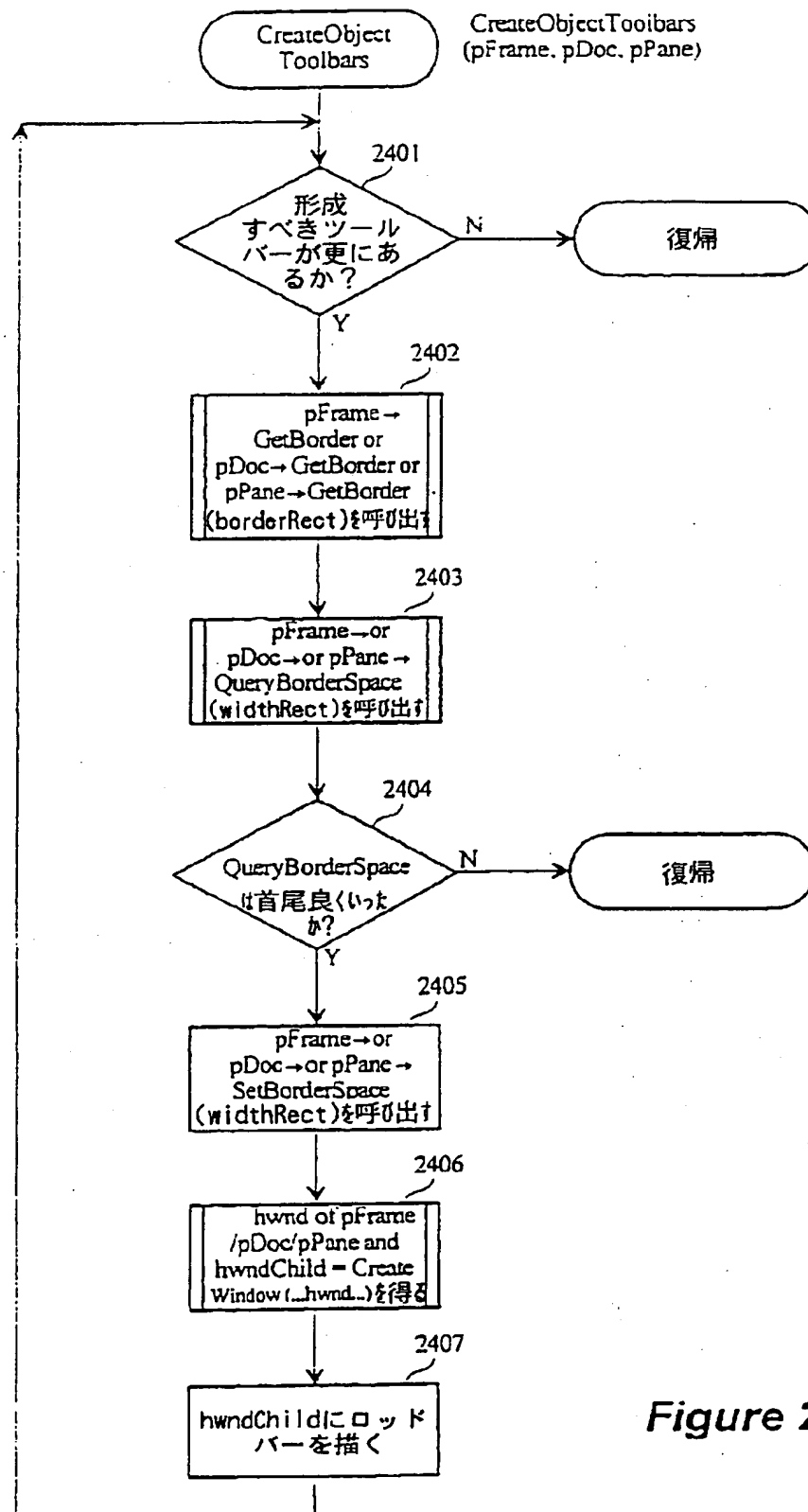


Figure 24

【図25】

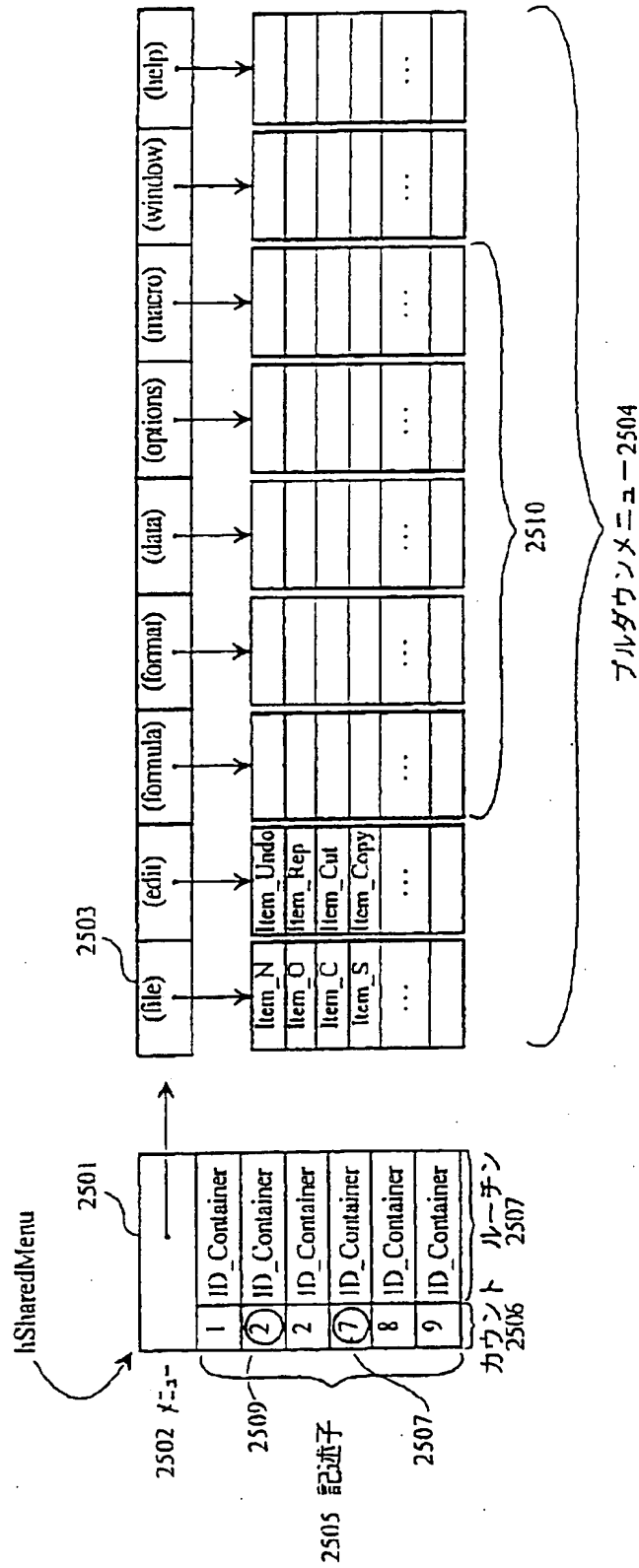


Figure 25

【図26】

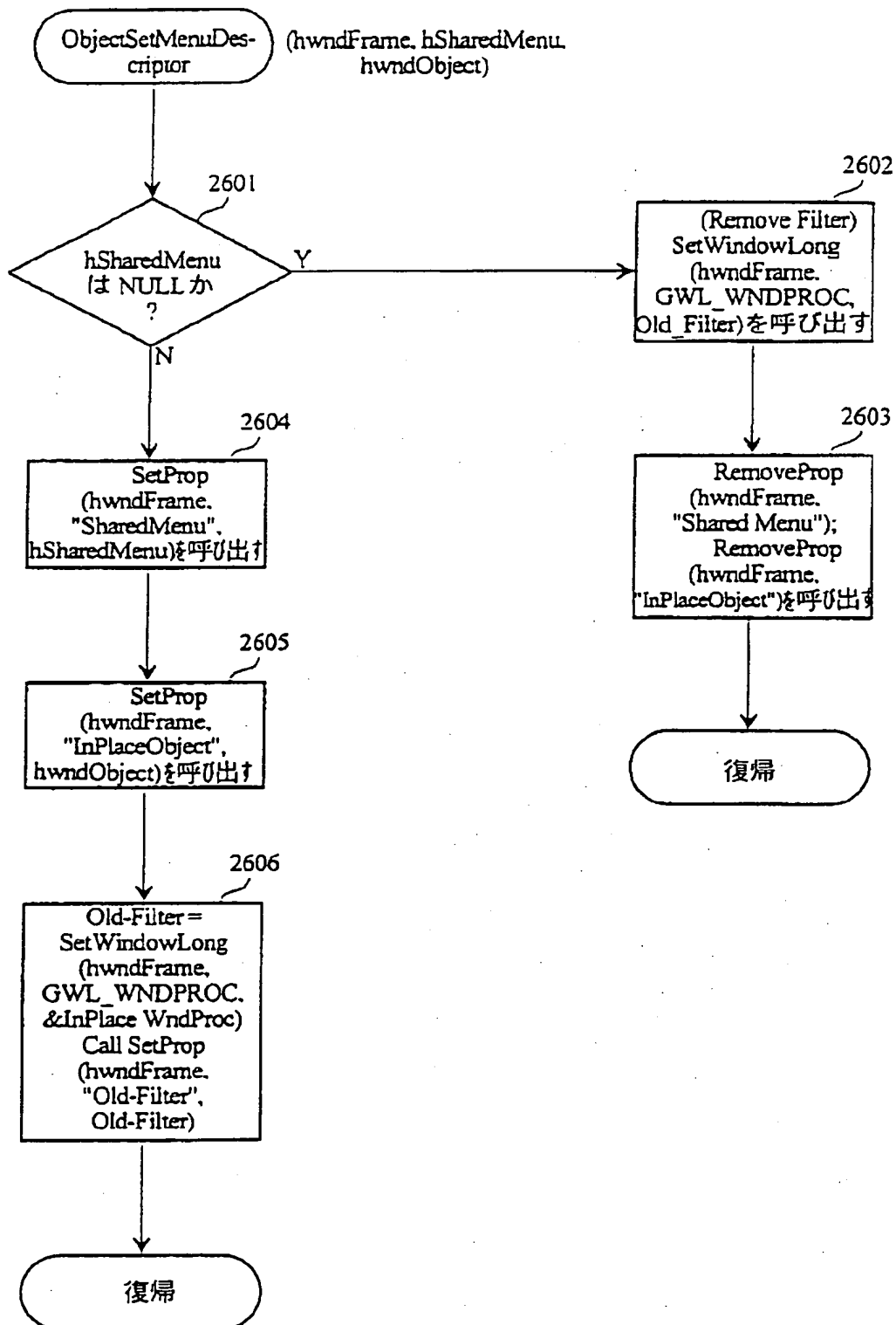


Figure 26

【図27】

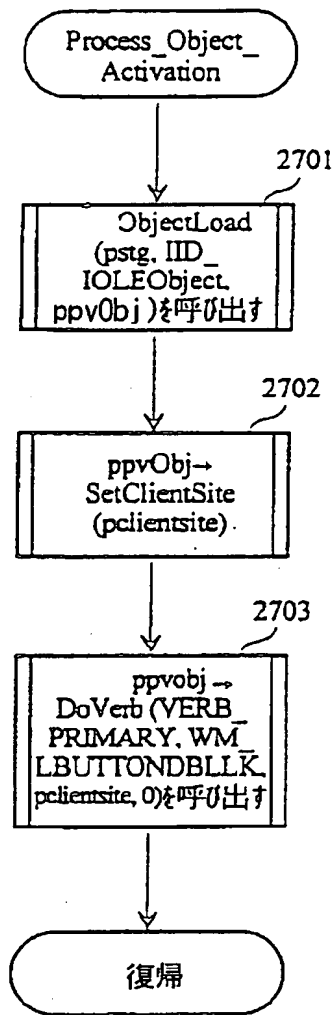


Figure 27

【図28】

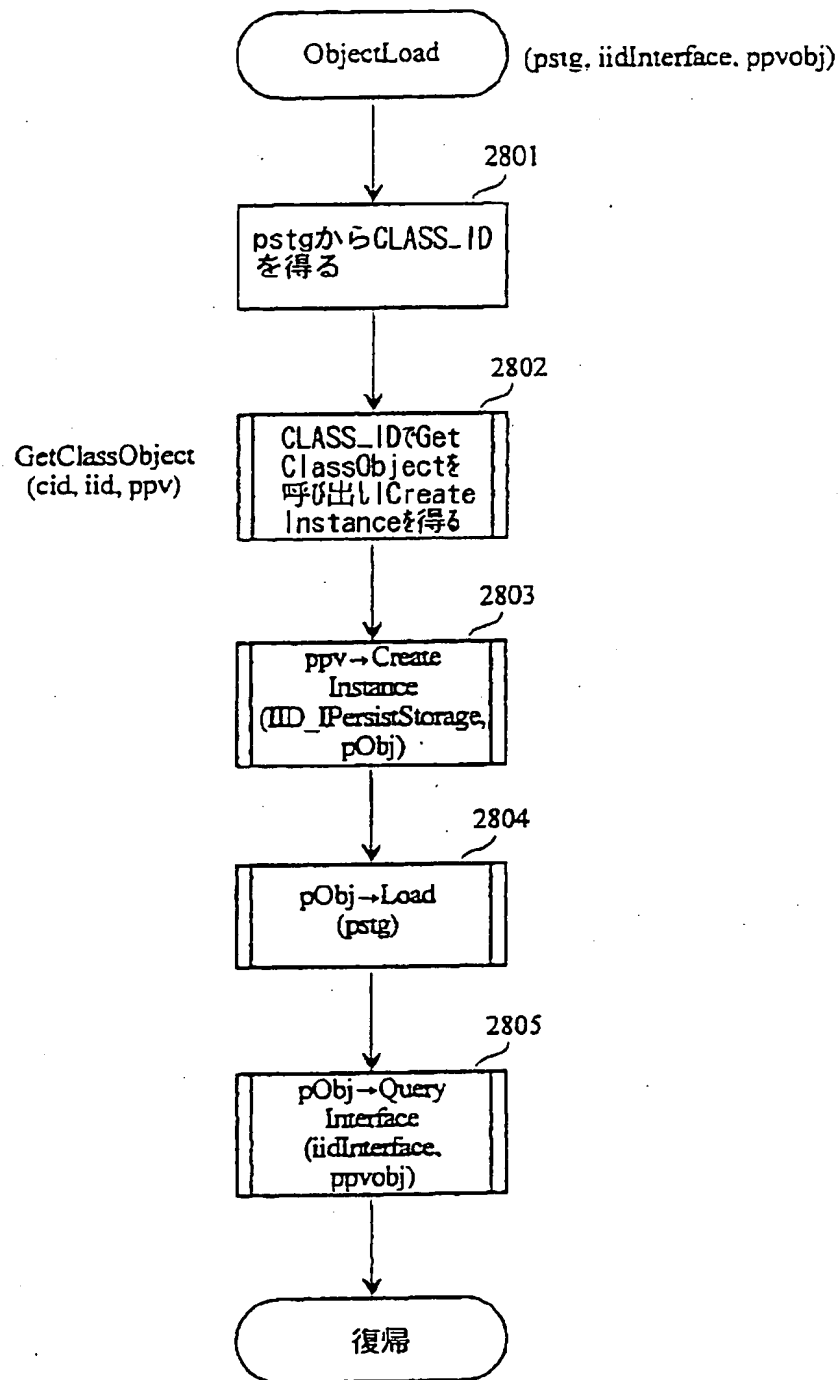


Figure 28

【図29A】

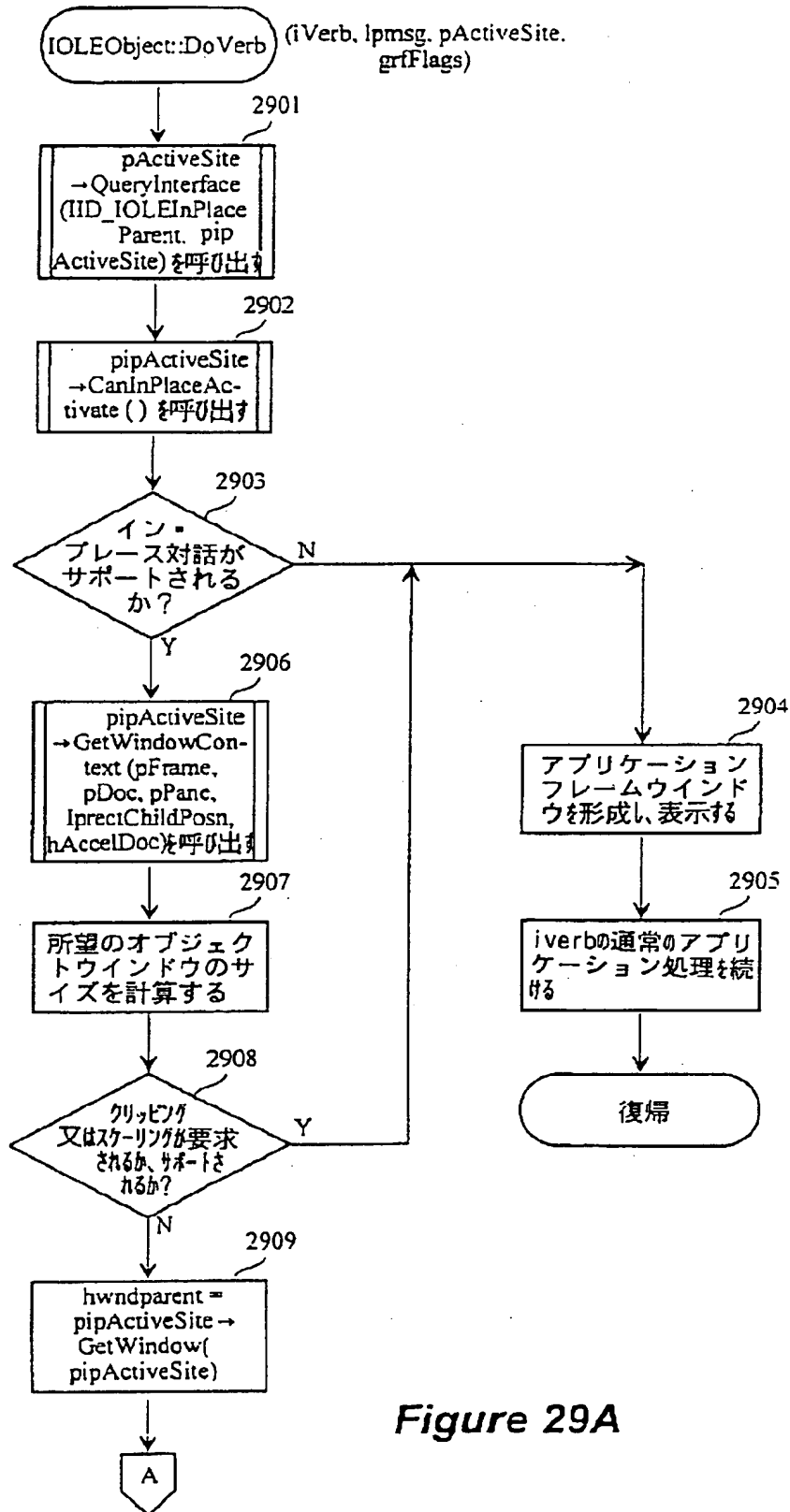


Figure 29A

【図29B】

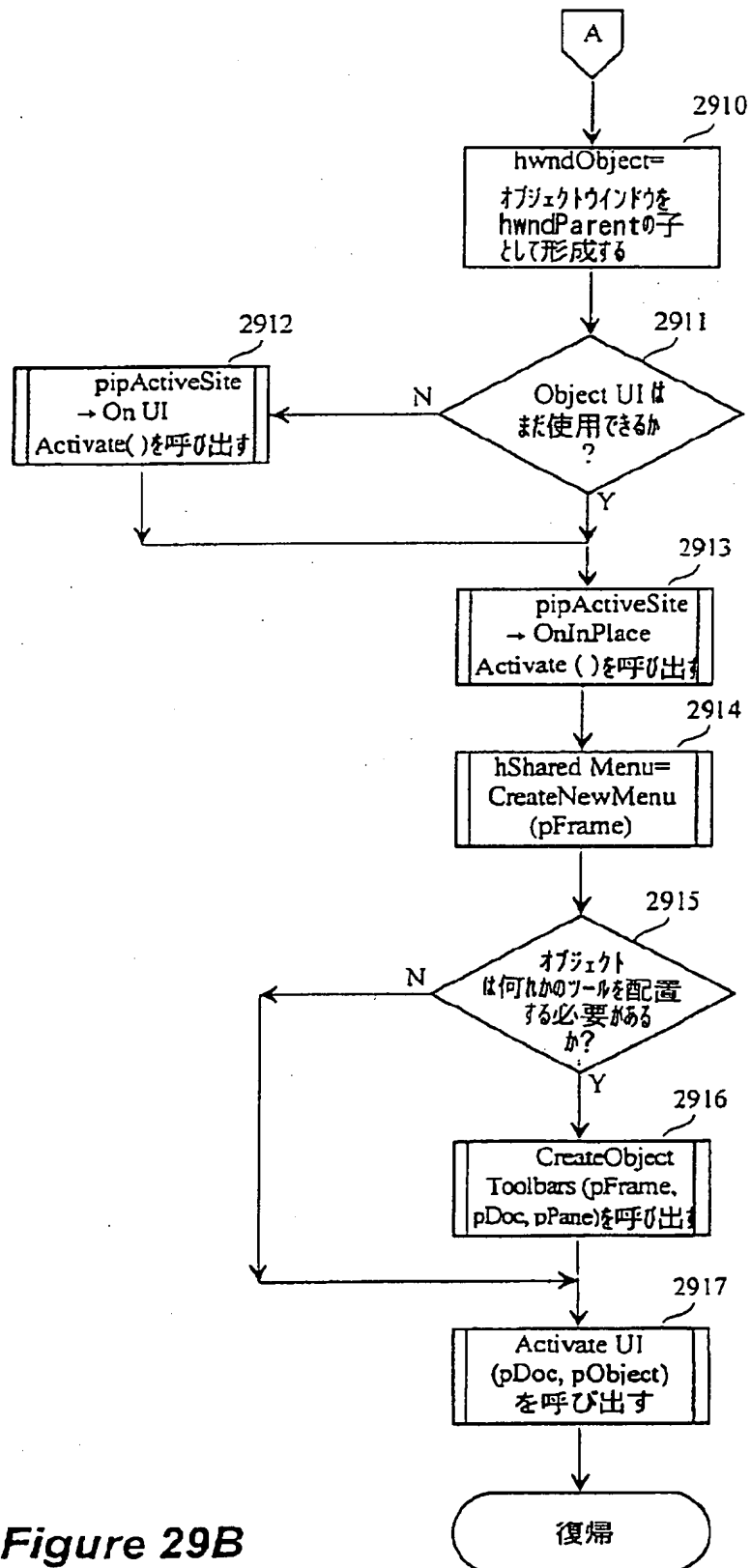


Figure 29B

【図 30】

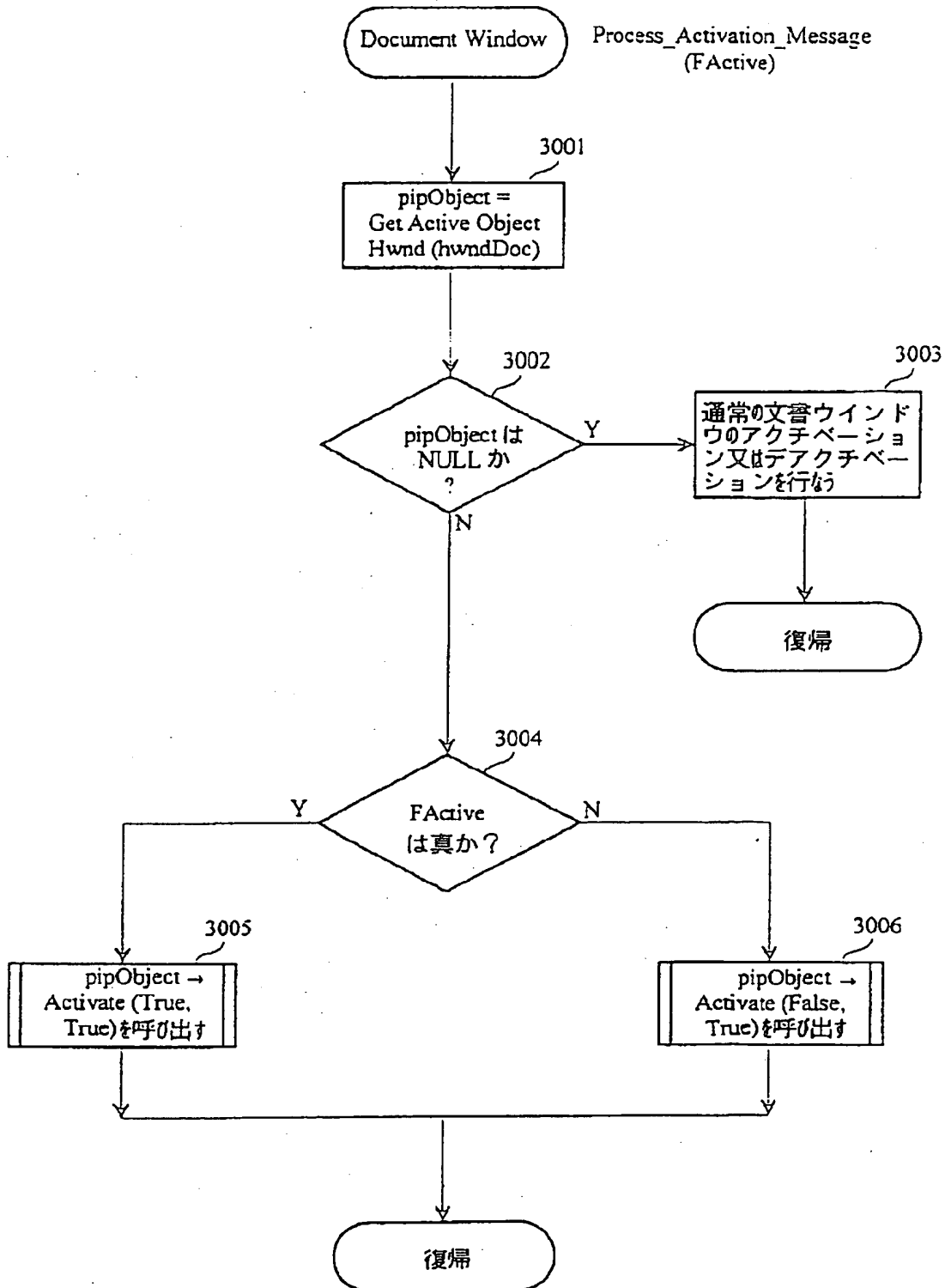


Figure 30

【図31】

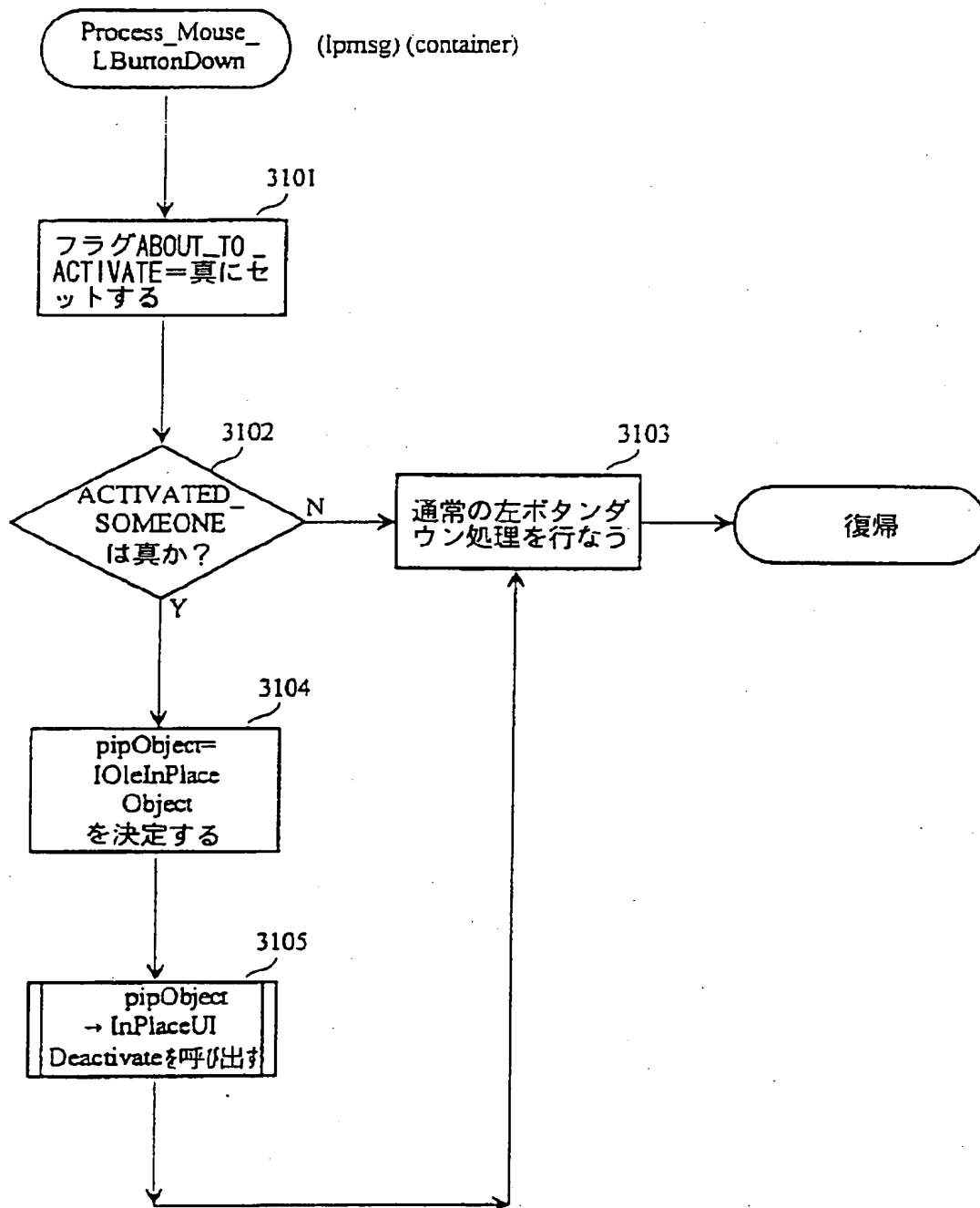


Figure 31

【手続補正書】

【提出日】1995年6月6日

【補正内容】

請求の範囲

1. コンピュータシステムにおいてコンテナオブジェクト内に収容されたコンテナオブジェクトをアクチベートするための方法であって、上記コンピュータシステムは、コードをスケジュールして実行するためのオペレーティングシステムを有し、上記コンテナオブジェクトは、コンテナウインドウ環境を伴うコンテナアプリケーションコードを有し、上記コンテナウインドウ環境は、コンテナオブジェクトと対話するためのコンテナリソースを有し、上記コンテナオブジェクトは、該コンテナオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションコードを有し、上記方法は、

上記コンテナアプリケーションコードを実行し、コンテナアプリケーションコードは、オペレーティングシステムによって個々にスケジュールすることができ、

上記コンテナウインドウ環境を表示し、

上記表示されたコンテナウインドウ環境内に上記コンテナオブジェクトを表示し、

上記コンテナオブジェクトをアクチベートし、そして

複数のサーバリソースを上記表示されたコンテナウインドウ環境内で一体化して表示し、上記サーバリソースは、サーバアプリケーションコードの一部分であり、サーバアプリケーションコードは、オペレーティングシステムにより個々にスケジュールすることができて、上記一体化されたサーバリソースの中からあるサーバリソースをユーザが選択するときに、サーバアプリケーションが個々にスケジュールされたエンティティとして実行されそしてサーバリソースの選択を処理する、

という段階を備えたことを特徴とする方法。

2. 上記コンテナアプリケーションコードは、このコンテナアプリケーションコードによって割り当てられて管理されるコンテナメニューを有し、上記サ

サーバアプリケーションコードは、このサーバアプリケーションコードによって割り当てられて管理されるサーバメニューを有し、複数のサーバリソースを一体化して表示する上記段階は、

複合メニューバーを発生し、そして

その発生された複合メニューバーにサーバメニュー及びコンテナメニューを記憶する、

というサブ段階を備えた請求項1に記載の方法。

3. 上記記憶段階は、サーバメニュー及びコンテナメニューを複合メニューバーにおいてインターリーブする請求項2に記載の方法。

4. 上記コンテナウインドウは、複数のメニューを表示するためのメニューバーを有し、そして上記複合メニューバーは、コンテナアプリケーションのメニューバーとして表示される請求項2に記載の方法。

5. 上記一体化されたサーバリソースの中からあるサーバリソースを選択するのに応答して、サーバアプリケーションコードを個々にスケジュールされたエンティティとして呼び出し、サーバリソースの選択を処理し、そして

上記表示されたコンテナウインドウ環境からあるコンテナリソースを選択するのに応答して、コンテナアプリケーションコードを呼び出し、コンテナリソースの選択を処理する、

という段階を更に備えた請求項1に記載の方法。

6. 上記コンテナアプリケーションコードは、データを表示するためのウインドウを上記表示されたコンテナウインドウ環境内に有し、そしてそのウインドウ内にサーバリソースを配置するようにコンテナアプリケーションコードと交渉する段階を備えた請求項1に記載の方法。

7. コンテナウインドウ環境内に複数のサーバリソースを一体化して表示する上記段階は、その表示を行う前又は行うときに、どのサーバリソースを表示すべきかをコンテナアプリケーションコードが知ることなく、行われる請求項1に記載の方法。

8. 上記コンテナアプリケーションコード及びサーバアプリケーションコード

は、単一のコンピュータプロセスにおいて個別のスレッドとして実行される請求項1に記載の方法。

9. 複数のサーバリソースを一体化した後に、上記コンテナウインドウ環境内のコンテナオブジェクトをハイライト処理して表示し、サーバリソースがユ

ーザ選択に使用できることを指示する段階を備えた請求項1に記載の方法。

10. 上記コンテナアプリケーションコードは、メッセージを受け取って処理するためのコンテナメッセージハンドラーを有し、上記サーバアプリケーションは、メッセージを受け取って処理するためのサーバメッセージハンドラーを有し、上記コンテナオブジェクトは、メッセージを受け取って処理するための現在メッセージハンドラーを有し、この現在メッセージハンドラー最初はコンテナメッセージハンドラーにセットされ、そして複数のサーバリソースを一体化して表示する上記段階は、コンテナオブジェクトの現在メッセージハンドラーを特殊なメッセージハンドラーにセットすることを含み、この特殊なメッセージハンドラーは、コンテナリソースが選択されたときにコンテナリソース選択メッセージをコンテナメッセージハンドラーへ送ると共に、サーバリソースが選択されたときにサーバリソース選択メッセージをサーバメッセージハンドラーへ送るものである請求項1に記載の方法。

11. 上記コンテナウインドウ環境はウインドウを有し、上記サーバアプリケーションコードは、ウインドウを伴うサーバウインドウ環境を有し、そして更に、

上記サーバウインドウ環境のウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

ユーザからメニューコマンドを受け取り、そして

上記メニューコマンドを受け取るのに応答して、

上記コンテナウインドウ環境のウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

メニューニューモニックを受け取り、そして

メニューニューモニックを受け取るのに応答して、サーバウインドウ環境のウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして

再指定する、

という段階を含む請求項1に記載の方法。

12. 上記コンテナウインドウ環境のウインドウは、フレームウインドウである請求項11に記載の方法。

13. 上記コンピュータシステムはキー入力のためのキーボードを有し、上記コン

テナアプリケーションコードは、コンテナリソースを選択するための複数のアクセラレーター組合せを有し、上記サーバアプリケーションコードは、サーバリソースを選択するための複数のアクセラレーター組合せを有し、そして更に、上記コンテナアプリケーションコードがアクセラレータキー組合せを受け取るときに、個別にスケジュールされたサーバアプリケーションコードを呼び出して、サーバリソースが選択されたかどうか決定する段階を備えた請求項1に記載の方法。

14. コンテナオブジェクトをアクチベートする上記段階は、

コンテナオブジェクトを選択し、そして

そのコンテナオブジェクトに対して実行されるべきオペレーションを指定する、

という段階を更に備えた請求項1に記載の方法。

15. コンテナオブジェクト内に収容されたコンテナオブジェクトをアクチベートするためのコンピュータシステムであって、このコンピュータシステムは、コードをスケジュールして実行するオペレーティングシステムを有し、上記コンテナオブジェクトは、コンテナウインドウ環境を伴うコンテナアプリケーションコードを有し、上記コンテナウインドウ環境は、コンテナオブジェクトと対話するためのコンテナリソースを有し、上記コンテナオブジェクトは、該コンテナオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションコードを有するようなコンピュータシステムにおいて、

上記コンテナウインドウ環境を表示する手段と、

上記表示されたコンテナウインドウ環境内に上記コンテナオブジェクトを表示する手段と、

上記表示されたコンテナウインドウ環境内のコンテナオブジェクトをアクチベートする手段と、

複数のサーバリソースを上記表示されたコンテナウインドウ環境内で一体化して表示する手段と、

上記一体化され表示されたサーバリソースの中からあるサーバリソースが選択されたときにサーバアプリケーションを個々にスケジュールされたエンティ

ティとして実行しそしてサーバリソースの選択を処理する手段と、

を備えたことを特徴とするコンピュータシステム。

16. 上記コンテナアプリケーションコードは、このコンテナアプリケーションコードにより割り当てられて管理されるコンテナメニューを有し、上記サーバアプリケーションコードは、このサーバアプリケーションコードによって割り当てられて管理されるサーバメニューを有し、上記一体化して表示する手段は、複合メニューバーを発生しそしてその複合メニューバー内にサーバメニュー及びコンテナメニューを記憶する手段を含む請求項15に記載のシステム。

17. 上記発生して記憶する手段は、サーバメニュー及びコンテナメニューを複合メニューバーにおいてインターリーブする手段を含む請求項16に記載のシステム。

18. 上記コンテナウインドウ環境は、複数のメニューを表示するメニューバーを有し、そして上記複合メニューバーは、コンテナウインドウ環境のメニューバーとして表示される請求項15に記載のシステム。

19. 上記表示されたコンテナウインドウ環境からコンテナリソースが選択されたときにコンテナリソース選択を処理するようにコンテナアプリケーションを実行する手段を更に備えた請求項15に記載のシステム。

20. 上記コンテナアプリケーションコードは、データを表示するためのウインドウをコンテナウインドウ環境内に有し、そして上記一体化して表示する手段は、そのウインドウ内にサーバリソースを配置するようにコンテナアプリケーションコードと交渉する手段を含む請求項15に記載のシステム。

21. コンテナウインドウ環境内に複数のサーバリソースを一体化して表示する

上記手段は、その表示を行う前又は行うときに、どのサーバリソースを表示すべきかをコンテナアプリケーションコードが知ることなく、行われる請求項15に記載のシステム。

22. 上記コンテナアプリケーションコード及びサーバアプリケーションコードは、単一のコンピュータプロセスにおいて個別のスレッドとして実行される請求項15に記載のシステム。

23. 上記コンテナウィンドウ環境内のコンテナオブジェクトをハイライト処理して表示し、サーバリソースがユーザ選択に使用できることを指示するための手段を更に備えた請求項15に記載のシステム。

24. 上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナアプリケーションコードは、コンテナリソースを選択するための複数のアクセラレータキー組合せを有し、上記サーバアプリケーションコードは、サーバリソースを選択するための複数のアクセラレータキー組合せを有し、そして更に、アクセラレータキー組合せを受け取るときにサーバリソースが選択されたかどうかを決定するためにサーバアプリケーションコードを個別に実行されるエンティティとして呼び出す手段を備えた請求項15に記載のシステム。

25. 上記アクチベートする手段は、コンテナオブジェクトを選択しそしてそのコンテナオブジェクトに対して実行されるべきオペレーションを指定する請求項15に記載のシステム。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER IPC 5 G06F9/44 G06F3/033		International Application No. PCT/US 93/11468
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) IPC 5 G06F		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practical, search terms used)		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BYTE, vol.17, no.14, December 1992, LONDON, GB pages 153 - 160 PETER WAYNER: 'Brave New Desktop' see page 155, right column, line 8 - page 156, middle column, line 2; figures ---	1-16, 22-34
A	'Microsoft Windows Version 3.1 User's Guide', MICROSOFT CORPORATION, US see page 487, paragraph 1 - page 489, last paragraph ---	1-16, 22-34
A	EP,A,0 215 203 (IBH) 25 March 1987 see the whole document --- -/--	1-16, 22-34
<input checked="" type="checkbox"/> Further documents are listed in the continuation of box C.		
<input checked="" type="checkbox"/> Patent family members are listed in annex.		
* Special categories of cited documents : "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier document but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art. "Z" document member of the same patent family		
Date of the actual completion of the international search 7 April 1994		Date of mailing of the international search report 29.08.94
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentmann 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 360-3016		Authorized officer Fonderson, A

INTERNATIONAL SEARCH REPORT

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		Inventor's Name PCT/US 93/11468
Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol.27, no.10B, March 1985, NEW YORK, US page 5998 'Object-Specific Command Bar' see the whole document	1-7, 14-16, 22-28
A	EP,A,0 304 071 (WANG LABORATORIES INC.) 22 February 1989 see abstract	1-16, 22-34

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US93/11468

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

SEE ANNEXED SHEET

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-16, 22-34

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest.
- ☐ No protest accompanied the payment of additional search fees.

INVITATION TO PAY ADDITIONAL FEES

International application No.
PCT/US 93/ 11468

1. Claims 1-16, 22-34 : Method and system of interacting with a contained object within the context of its containing application.
2. Claim 17 : Method for dynamically combining window hierarchies in a computer system.
3. Claims 18-20 : Method for scrolling a window containing another window which contains selected data.
4. Claim 21 : Method for visually emphasising a selected object which is displayed on a display device.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No.

PCT/US 93/11468

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-0215203	25-03-87	US-A- 4815029	21-03-89
		JP-A- 62072059	02-04-87

EP-A-0304071	22-02-89	AU-B- 607795	14-03-91
		AU-A- 2094388	23-02-89
		JP-A- 1126736	18-05-89
		US-A- 5206951	27-04-93
		US-A- 5261080	09-11-93
		US-A- 5303379	12-04-94
		US-A- 5226161	06-07-93

フロントページの続き

(51)Int.Cl. ⁶	識別記号	庁内整理番号	F I	
		9288-5 L	G O 6 F 15/20	5 8 0 J
(72)発明者	マッキチャン バリー ビー			
	アメリカ合衆国 ワシントン州 98110			
	ベインブリッジ アイランド マンザニタ			
	ロード ノースイースト 12730			
(72)発明者	マックダニエル リチャード			
	アメリカ合衆国 ペンシルバニア州			
	15206 ピッツバーグ スタントン アベ			
	ニュー 6017 アpartment 1			
(72)発明者	レマラ ラオ ヴィー			
	アメリカ合衆国 ワシントン州 98072			
	ウッディンヴィル ノースイースト ワン			
	ハンドレッドアンドフィフティファースト			
	ストリート 19011			
(72)発明者	ウィリアムズ アントニー エス			
	アメリカ合衆国 ワシントン州 98053			
	レッドモンド ノースイースト フォーテ			
	イシックス ストリート 22542			

【要約の続き】

ス選択を処理する。

THIS PAGE BLANK (USPTO)